44 MEMEX (SEE EARLIER ISSUE)

THE BEST SELLING COMPUTER PROJECTS MAGAZINE **AUGUST 1984**

ELECTRONICS & COMPUTING

EMAP PUBLICATION 12 22 REF 24 REF 34 REF 37

40 CENTRONICS TO RS232 CONVERTER 47 52 60

USA $2.95
Germany D5.80 **90p**

EXCLUSIVE
FIRST ACORNSOFT PRE-LAUNCH OFFER
20% OFF VIEW UTILITY CASSETTE
TO THE FIRST 300 READERS

FROM PALETTE TO PIXEL

A guide to micro graphics aids

SINCLAIR IN THE DOCK-QL AND THE VER$ STORY

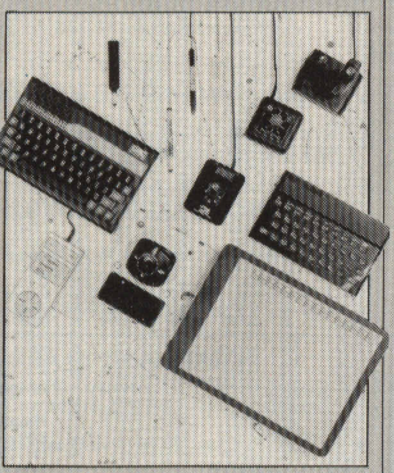
MEMEX AT LAST·BUILDING DIY ROBOTS

BEEB VOLT VIEWER–AN ADC GRAPHIC DISPLAY

SPECIAL OFFER
Acornsoft VIEW Printer Driver Cassette
see page 63

ELECTRONICS & COMPUTING Contents

Vol. 4 Issue 8



COVER PHOTO BY ROB BRIMSON

Electronics & Computing Monthly
Scriptor Court, 155 Farringdon Road,
London, EC1R 3AD

Editorial 01-833-0846
Editor Gary Evans
Assistant Editor William Owen
Production Editor Liz Gregory
Administration Serena Hadley
Legal Editor Rufus J. Hackenbush

Advertising 01-833-0531
Advertisement Manager Anthony Herman
Chief Executives Richard Jansz
Francis Lee
Classified Tracey Keighley
Assistant Yvonne Moyser

Production 01-833-0531
Art Editor Jeremy Webb
Make-up Time Graphics
Publisher Alfred Rolington

Distribution
EMAP National Publications

Published by
EMAP Business and
Computer Publications

Printed by
Riverside Press, England

Subscriptions
Electronics & Computing Monthly,
(Subscription Department),
Competition House, Farndon Road,
Market Harborough, Leicestershire.

Electronics & Computing Monthly is normally published on the 13th day of each month.

Reasonable care is taken to avoid errors in this magazine however, no liability is accepted for any mistakes which may occur. No material in this publication may be reproduced in any way without the written consent of the publishers. Subscription rates: UK £11.30 incl. post. For overseas rates apply to Subscription Dept., Competition House, Farndon Road, Market Harborough, Leicestershire. Back issues available from: EMAP National Publications (*E&CM* Back Numbers), Bretton Court, Peterborough, PE3 8DZ. Phone: 0733 264666.

PROJECTS

FEATURES

REVIEWS

PLUS

And within *Your Robot*

How to build robots from LEGO bricks. The first article describes a wall builder. Plus the final part in the Motors explained series.

The Beeb copies big blue

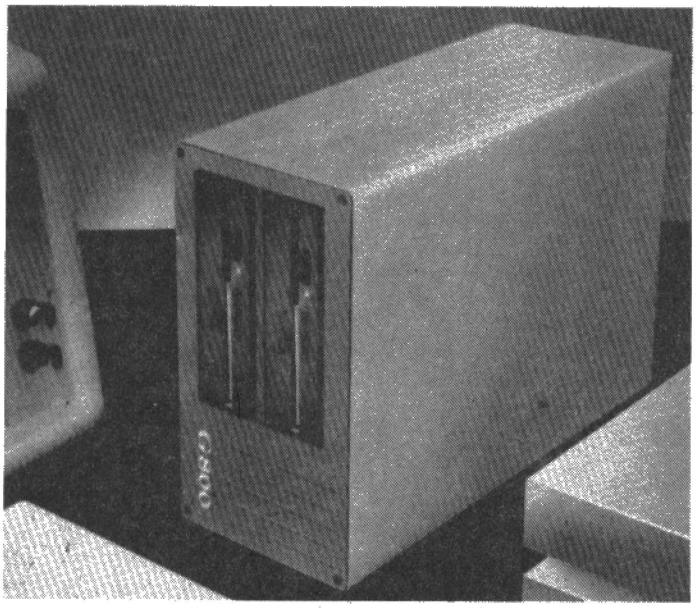
Torch Computers are surely one of the most prolific third party (no, we should say first party – Acorn took them over last month) supporters of the BBC micro in the business. First we had their Z80 disc processor, followed by a simple Z80 board, then UNIX on a 68000 processor with dual disk drive, and now the Graduate – an IBM compatible 8088 with 128 or 256K RAM and PC compatible disks.

Graduate was designed by Data Technologies under licence from Acorn. Top of the range is the G800/2 which sells at £999. It provides a 256K 8088, twin 320K PC compatible disk drives, two IBM compatible hardware expansion slots, MS-DOS upgraded to PC compatibility, and of course a user manual. The alternative is a single drive unit for £764.

The high degree of compatibility achieved is indicated by the claim that both Lotus 1-2-3 and the Microsoft flight simulator will run on the Graduate. Both these programs are notoriously hardware specific, relying heavily on IBM screen calls.

Clearly the BBC is now living up to its reputation as a versatile terminal, supported by a wide range of processors. The IBM package from Torch is likely to become very popular, offering a dual disk drive IBM system for £1400 – much cheaper than big blue's machine but minus a monitor. Perhaps more surprisingly, Torch's expensive £3000 UNIX system is already selling well, with deliveries to customers exceeding 30 per month.

The inadequacies of the Sinclair Spectrum have spawned yet another add-on; in this case a 3 channel sound synthesiser from DK'tronics to make up for the computer's poor sound quality. The synthesiser, pictured above, allows Spectrum users to program their own music, explosions, zaps, chimes, whistles and harmonies. Also available is a beep amp, which is a high power audio amplifier which improves the sound quality of the Spectrum beep. The synthesiser is priced at £29.95, and the amplifier at £14.95.



The Model A is dead

Acorn have hammered the last nail into the coffin of the Model A, the stripped down version of the BBC micro which they believed (incorrectly) would sell in bigger numbers than the model B.

Over 300,000 BBC micros have been sold since the launch in 1982, but only a few of these were Model As, with its smaller memory (16K) and fewer interfaces. The Model A was in effect replaced by the Electron in late 1983, but Acorn have waited six months to declare the death official. There are unlikely to be many mourners at the funeral, but if you really care then just type *FX254,0 into your Model B, and it will become a Model A reborn. (PS. you have to switch off the power to get out of this one).

Setting the tone with a laid back synthesiser

Renault, the car manufacturer, is the latest Company to incorporate a speech synthesiser as part of the instrumentation on their new range of cars.

They have obviously taken a great deal of care over the specification of the speech generator, deciding that it must neither sound like a back seat driver, continually nagging, nor like a policeman giving warnings about unfastened safety belts or speed limit violations! The press relieease goes on to state that, by the end of the exercise, the Company had a system that was so laconic that it was necessary to include a Test button which permits the driver to listen to all the available messages at least once. It fails to mention what the messages' final decision was, so if you want to know take a trip to your nearest Renault dealer.

Wonder what the chances are of *E&CM* having a car for review!

The DFS which doesn't pinch the RAM

The 3K stolen from RAM has always been an irritation to BBC DFS users – but the itch has now been scratched.

A company called MRM have introduced the first no-RAM DFS which will work on any BBC without second processor or sideways RAM. The unit is called the EOO DFS (because instead of throwing up the E00 page number it gives the higher hex number of 1900). The ROM comes on a simple PCB with two chips (one is extra RAM) and plugs into one of the usual ROM sockets with a lead connected to one IC within the Beeb.

The result is that users can now use those cassette programs which are too large to run alongside a normal DFS.

The device costs £24.95, is DFS compatible, and Acorn approved.

Disk drive survey

Last month's survey of the disk drives available to home users unfortunately omitted any reference to the Spider Systems 3" disk drives. The drives are based on a Hitachi design, adapted to home use with a strong aluminium extrusion finish. Double sided, single or double density versions are available, and are compatible with both the BBC and Spectrum micros. The double density version retails at £219.99, (including a utility disk), and the single sided version at £179.99. Both prices are all inclusive.

Conflict and contradiction

Compare and contrast: 'The Commodore 64 is now the best-selling home computer in the UK, according to the latest figures compiled by Research Analysis Marketing. Commodore toppled Sinclair's Spectrum from the top slot in March and the Commodore 64 continues to head RAM's list of the top-selling micros in the under £1000 bracket'. and . . .

'Britain's Sinclair Research continued to dominate the UK personal computer market during the first quarter 1984, achieving 43% of unit sales – according to independent market research organisation, Audits of Great Britain Ltd . . . AGB gave second and third place to Commodore (28%) and Acorn (10%). Sinclair's ZX Spectrum remained the most popular model with an individual 36% share, up 2% from fourth quarter 1983'.

It's nice to know that in computers at least everybody comes out the winner.

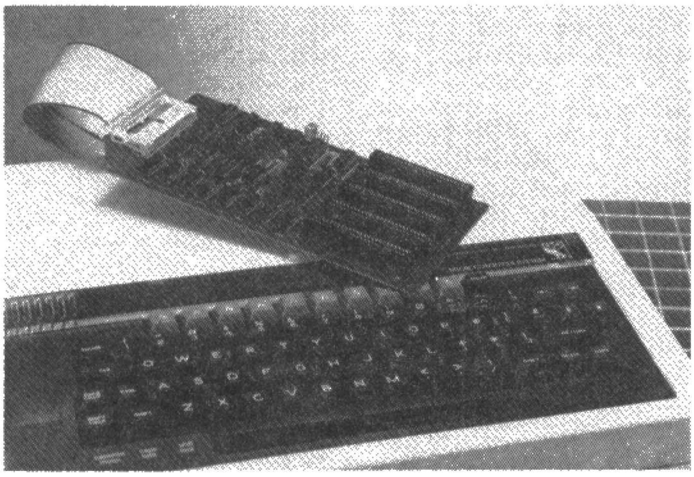
Expanding the price

Anyone wondering whatever happened to the Acorn Electron should tune into Channel 4 to watch the inteminable adverts for 'the micro that speaks the same language as the BBC computer' or words to that effect.

The advert is aimed at the computer illiterate parents of bratish hackers; what it doesn't tell you is that the upgrade package containing printer, joystick and ROM cartridge interfaces costs a phenomenal £59.99, slamming the Electron way up in to the £260 price bracket.

The Plus-1 expansion unit 'upgrades the Electron to a practical home office computer and versatile games and programming machine' so the PR blurb tells us. It is such a pity that potential buyers are not told that the bare £200 computer is (by implication) impracticable.

A slightly cheaper alternative is the FBC Systems' First Byte printer interface. Users with no interest in joystick or ROM cartridge ports can obtain this expansion unit (whether it is serial or parallel is not specified) for £34.95. The interface is said to require no software driver, either on tape or within EPROM, and, because of the Electron's limited power supply is of a very low power design. *FBC Systems, 0332 365280.*



Prize for the most ingriguing product of the month goes to the XMEM backplane (pictured above) for the BBC micro. The board is manufactured by Xcalibur Computers; it has five slots and 64K of RAM, and plugs directly into the 1MHz bus, taking power from the BBC auxiliary connector (for disk users, an external power supply is available). Best feature of all in this unit is the ability to accept Apple II I/O cards – of which there are an almost infinite number – which can be run directly from BBC BASIC or machine code. In use, the input/output addresses of the Apple are mapped into defined addresses on the BBC, which can then drive the card(s) through BASIC commands or assembler. All Xcalibur cards are compatible and are documented in the XMEM manual; these include A/D and D/A converters, multi-channel counters, multi-channel port communications, graphics processors, calendar/clocks, plotters, digitizers, bar code wands, parallel interfaces etc. The unit is priced at around £250; bare boards are available at a cheaper price, or fully packaged boards with external power supply for slightly more. *Xcalibur Computers, Spencer House, 3 Spencer Parade, Northampton NN1 5AB.*

New languages

The turtle graphics language Logo is now available via Sinclair for the Spectrum. The package is priced at £39.95 and includes an extensive user guide.

Logo adopts a 'learning through experience' approach to education, encouraging curiosity and avoiding restrictions. Thus, it will interpret rather than reject a bad instruction.

Central to the language is the graphics 'turtle', a screen cursor controlled directly by the user in simple steps by direction and length of movement.

Acornsoft have also extended the range of languages in their catalogue, with four new titles for the BBC micro: S-Pascal, Turtle Graphics, LISP Demonstrations, and BCPL calculations.

Acorn promotion

Acorn, rumoured to be prevented by the BBC from lowering the price of the BBC micro, have adopted a compromise solution by giving a free data recorder plus £80 worth of software with every micro sold in July.

Customers who do not require a data recorder will be allowed to choose three more Acornsoft programs instead.

A competitor for Microdrive

Another new complication in the disk/cassette conundrum has appeared on the Spectrum horizon: it is Wafadrive, a fast storage £129 device with a capacity of up to 128K (a pound a kilobyte).

Wafadrive features two tape drives. The retrieval time is not specified but claimed to be fast. The 'wafer' cartridges are available in 128K, 64K and 16K – the smaller the storage capacity the faster the memory – and priced between £3.95 and £3.45.

The Wafadrive has both RS232 and Centronics compatible outputs, allowing the computer to drive the majority of popular printers available. A specially written word processor is included in the package, and several applications and games software titles will be available shortly. The package also includes extended BASIC commands made available by the Wafadrive Operating System.

Versions of the Wafadrive will be available for the Commodore 64 and VIC 20 in the near future.

Software update

Friendly Face is a new interface program from Monitor which is capable of converting programs for running on the Microdrive. The cassette version retails at £6.95 whilst the cartridge includes 'tasmerge' and 'mf merge' which enables it to merge with the Tasword Two and Masterfile programs. It is approved by Tasman Software and Campbell Systems respectively and the Microdrive cartridge version costs £12.95. A useful new product from Number One Systems is **Analyser** for the Spectrum 48K which is helpful in saving time on bench testing as it tests amplifier and filter circuits. Other products for both the 16K and 48K machines include Timedata's **Screen Enhancer,** a machine code program which facilitates up to 32 lines of 64 characters and **Moder-80** which is a disassembler with an additional mini-assembler. The monitor features include breakpoint, copy and hex arithmetic and conversion. It is available by mail order only from Seven Stars Publishing, 15 Gloucester Avenue, London NW1 7AW.

For the Commodore 64, Honeyfold Software have introduced **Honey.Aid2** which is a utility program adding 100 new commands to CBM BASIC. These are toolkit commands including APPEND, EDIT, OPTION, STORE, QDOS with many others and also extended BASIC commands. Logic 3 have announced a multi-tasking version of **FORTH** which comes in disk format with a 40 page manual. Priced £29.95 it has the stimulating title of **Logic 3 Forth.** Mirrorsoft have introduced **Go-Sprite** which is a sprite editor which may be operated simply by a joystick with optional lightpen and keyboard control. Facilities include a choice of up to 7 layers of overlay with sprite data files on disc or tape and data display option.

Finally, BBC owners who have a particular interest in graphics may be interested in two new software packages introduced to enhance the PL graphics system, which itself retails at £130.95 plus VAT for domestic users. **Decoder** allows the decoding of pictures into X and Y co-ordinates which then may be screen-listed, possibly in blocks at a time and which may be put out to a printer. **Quest** provides facilities for data analysis and assessment with the optional facility of designing questionnaires. Both programs retail for around £10.00 plus VAT. Cases Computer Simulations have announced that their **Manage** game which is used for management training is now available for the BBC B as well as the Spectrum 48K.

Copyright law in confusion

The software industry is justifiably concerned about the spread of piracy. However, the subsequent reaction has affected the innocent as well as the guilty, and the freedom of the computer press to inform its readers is threatened. Gary Evans has set out to put the law into perspective.

The current copyright and patent laws do not as yet seem to acknowledge the fact that computer software exists and this lamentable state of affairs is causing many organisations, among them *E&CM,* more than a few problems.

While anyone with the interests of the industry at heart would agree that some form of protection against wholesale piracy of software is in the best interests of all concerned the present legal framework in this country makes is inadequate.

Traditionally, the way of protecting mainframe software was by way of contract law, in that one of the terms of the sale was that the purchaser would undertake not to make any copies of the software. Any breach of these conditions would leave the purchaser open to legal action in that they were in breach of contract.

In the case of most micro software, the pirate is often in law considered to be a third party, and as such no contract exists between the pirate and the vendor of the software. In this case pursuit of the pirate under the law of contract cannot apply.

For this reason many legal experts in this country have turned to the copyright laws to protect the interests of the software producers. The act on which current law is based is the Copyright Act of 1956. Not surprisingly, this act makes no mention of software or computer programs as thirty or so years ago there was no cause for the legislature to concern itself with computers. Not so today.

The sections of this act that are seen to apply to software come within the clauses referring to work that may be classified as literary, dramatic, musical or artistic. Software is not known to be musical although some may claim that it could be seen as being artistic and even dramatic on occasion – it is however under the heading of 'literary work' that most actions brought to date have been pursued. Programs published in the ink and paper medium would, by common consent be covered by this section of the act but quite how this 1956 act would apply to software in machine readable form is not so clear cut.

A Government Green Paper published in July 1981 recommended that 'explicit new legislation' should be introduced in order that programs should be capable of protection under the same conditions as literary works. The same paper goes on to suggest that copyright should extend to software fixed in any form in which it may be reproduced.

Needless to say the recommendations of this report have – to date – not been implemented and the situation is still unresolved.

'software copyright is based on a 30-year-old law'

The legal system of this country then seems unable to offer a clear cut route by which organisations that have invested large amounts of time and money in developing software can protect their interest. With the hysteria that often surrounds the subject of software piracy it is not surprising that in some cases companies are becoming almost paranoic when it comes to the reproduction of software. It is quite usual to find copyright messages of the following form lurking inside the documentation that accompanies today's software packages:

> "Neither the whole or any part of the information contained in, or the product described in, this document may be reproduced in any material form without the prior consent of the copyright holders"

In some cases the warnings go further and proclaim that the software referred to may not be modified in any way. To put this in context, this form of message is suggesting that having bought a particular piece of software the purchaser is restrained not only from copying same for his own use but also from making any changes to the code to modify it for his own application.

This whole state of affairs is thrown askew when it comes to the case of disk based software. Most suppliers of such software conceed that it is desirable to make back-up copies in order to avoid a master disk being corrupted. In this circumstance there is an implied licence granted to the user such that they may copy the software onto their own disk. Indeed documentation that accompanies the majority of disk software positively encourages the creation of a back up disk.

There seems a very tenuous distinction between the form in which software is fixed. In the case of disks although the usual copyright protection would apply to anyone making copies and distributing them for profit there is the implied licence allowing individuals to make a copy. In the case of ROM or EPROM based software the rules, as to implied licence, seem to be different. Although, to our knowledge, no such ase has been brought to court, our best advice is that the matter of such an implied licence would at best be a point of contention. To our mind this is a nonesense!

The arguments that dictate that it is prudent to produce backups of disk based software rest on the corruptable nature of the media. The same arguments apply equally to ROM based software. We are all aware that ROMs are easily damaged by static electricity and it could well be argued that a prudent person, given the free use of an EPROM programmer would be well advised to produce a backup of any valuable software stored in this form. The reason that this practice is not more common is solely due to the fact that not too many computer users have easy access to an EPROM blower.

This argument would have to be tested in a court of law but seems to us to have a sound basis in reason. The only problem about testing such a point in law is that it could cost a great deal of money to prove the point.

The question of an implied licence in respect of software other than disk based material is at least touched upon by one major manufacturer. Dragon Data, in the case of the cassette based version of their 'Alldream' software encourage users to dump the software to EPROM in order that it may be used more effectively. Thus Dragon, in our opinion, are granting an implied licence to users of 'Alldream', allowing them to copy the cassette based software to an alternative medium for their own use, although once again the company would pursue any duplication of the product for gain.

'it is not surprising that some companies are becoming paranoic'

Publications such as *E&CM* are in a difficult position given the current state of the law. While the question of an implied licence in respect of either cassette based or semiconductor based software remains open we will remain open to action if we publish any material that can be said to encourage copying. This restriction applies to any software that may show how to accomplish such copying and may in some cases extend to mere references to copying.

In the case of ROM based software this situation is clearly ludicrous. Anyone wishing to make a copy of ROM based software in an EPROM must have access to an EPROM blower. Assuming that this is the case the simplest way to make copies of the software would be a straight copy of the ROM without any involvement from a micro. Indeed most blowers would support a direct copy mode it stands to reason that any 'pirate' would adopt this approach. Publishing the means by which ROM software may be loaded into a computer will be of interest only to those individuals who wish to examine and modify the software for their own use. We are however prevented from publishing such details and so are restrained from offering our readers the means by which they can further explore the use of their computer system. It is not the 'pirates' that lose out from this sort of attitude but those people who wish to use their computers for creative and educational purposes – the people that *E&CM* aim to serve.

The vulnerability of magazines to the type of legal redress that seems to be popular with the computer industry at the moment was demonstrated by the recent PCW case. Here the publishers of Personal Computer World had an interlocutory injunction taken out against them forcing the withdrawal of all the issues on sale at newsagents throughout the country. In order to avoid 'losing the issue' with the consequential loss of revenue, the publishers were obliged to accept an out of court settlement that, although punitive, was the lesser of two evils. The final irony in this case was that the material that formed the basis of the court action contained errors which meant that it would not perform the copying action as described, but of course it could be seen as in inducement to copy protected software.

'we are restrained from offering our readers the means by which they can explore their computer'

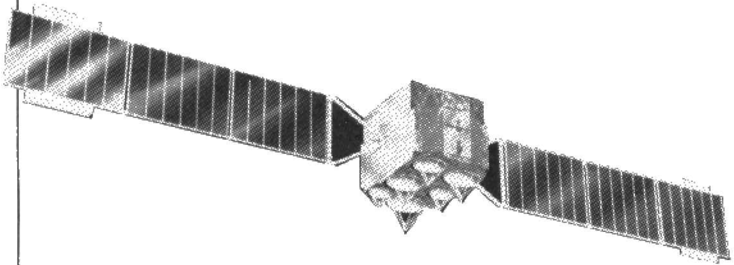
Among recent discussions of the implied licence issue a number of interesting points have come to light. For example what is the situation with ROM based software that, during normal use, requires that sections of code are loaded into RAM. In this case surely there is an implied licence allowing copying as this is a necessary action of the program is to function as intended. In this case there is likely to be no mention of any such implied condition yet it undoubtedly exists. If it exists in the case of some ROM software then why not all? Also just how much copying must go on before the question of an implied license would arise – many programs would call for the odd few bytes to be copied to RAM and surely once the concept of copying just one byte is acknowledged then an implied license could have been said to be granted.

The above clearly illustrates the current disarray in this area of the law, a situation that needs to be addressed in the very near future.

The paranoia of the computer industry now seems to be reaching beyond mere copyright on computer software and increasingly the law of patents is being applied to both software and hardware. Here again the courts are hardly the place to sort out conflicts that arise. Legal action is often prohibitively expensive for all parties concerned in a dispute, and the judiciary are often ill equipped to deal with the technical points raised in the course of an action. While meaning no disrespect to the Judges of the land, many of these erstwhile gentlemen where well past adolescence when the valve was invented and can hardly be au fait with the latest generation of computers.

It is high time that the law as regards copyright and patent of computer soft and hardware were explicitly covered by some form of legislation. It might also be beneficial to the industry at large if some form of tribunal could be formed to sort out conflicts in this area. This has worked reasonably in the field of industrial relations providing a low cost route by which disputes can be resolved. How about a similar body to deal with the problems of the computer industry?

NEXT MONTH IN E&CM



Acts of God and interlocutory injunctions permitting the September issue will be on sale on August 13th.

Communications special

We A major review of various aspects of the communications industry, in particular examining how advances in this area will affect microcomputer users over the next decade.

Major features will include

- A look at computer databases including Micronet, Prestel and the new Compunet service.
- How radio amateurs are using the air waves for a variety of computer related uses.
- Developments in the commercial sphere that are likely to filter down to micro users in the near future.

Spectrum word processor

Turn your Spectrum into a low cost word processing system. Full software listing and detailed description of operation all in next month's issue.

Making the most of allophones

Many speech synthesis systems today are adopting the allophone approach to producing speech. As anybody who has used such a system will know, creating even simple words can be rather a protracted process involving a great deal of trial and error. Our article next month provides a guide to the creation of allophone-based words with plenty of practical examples.

Spectrum frequency meter

Turn your Spectrum into an accurate frequency meter covering the range 00 to 65535Hz!

Plus all our usual BBC features and the latest news from the home micro market.

BRITAIN'S BEST SELLING COMPUTER PROJECTS AND SOFTWARE APPLICATIONS MAGAZINE

THE VOLT VIEWER

Mike Williams shows you how to monitor voltages in real time at each of the four ADC inputs of the BBC micro – without a single tacky voltmeter in sight.

At the back of the BBC Micro is a D-type socket into which games addicts plug their joysticks. For the experimenter, this port presents the opportunity to monitor simultaneously up to four different DC voltages (of not more than 1.8V each) through the BASIC command ADVAL(channel). This command returns a number related to the voltage applied to the relevant pins of the ADC port.

This program converts the voltage at each of the channels into a constantly updated bar chart. Each bar corresponds to one of the four channels and is in a different colour. Despite the ease of BASIC, it was simply not fast enough to cope with rapidly changing voltages on four different channels, hence the need for a machine code bar plotting routine shown in **Listing 1.**

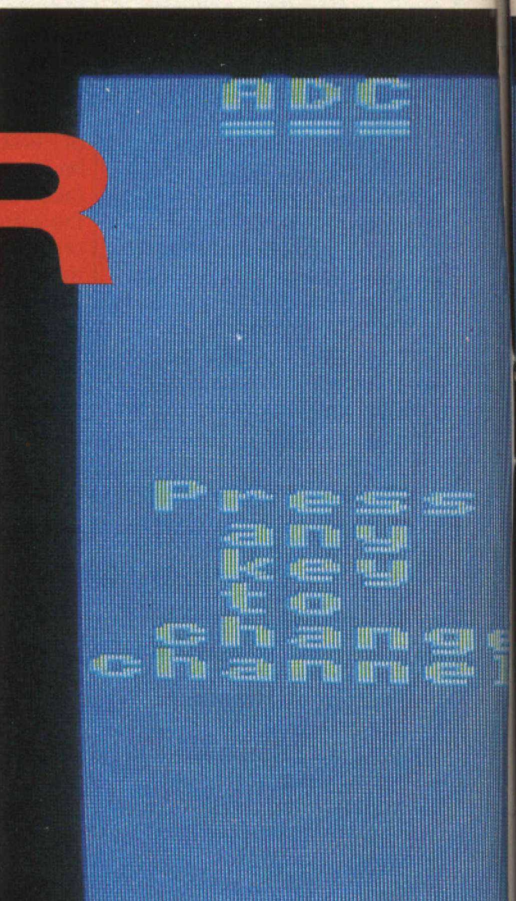
ADC2 creates a text window for running a new program.

If you are not interested in how this works then you can ignore the next couple of paragraphs and start typing in the listing. (The idle rich can write to me for the program on tape or disc).

The code is assembled in PROCassemble. It consists of two main subroutines:

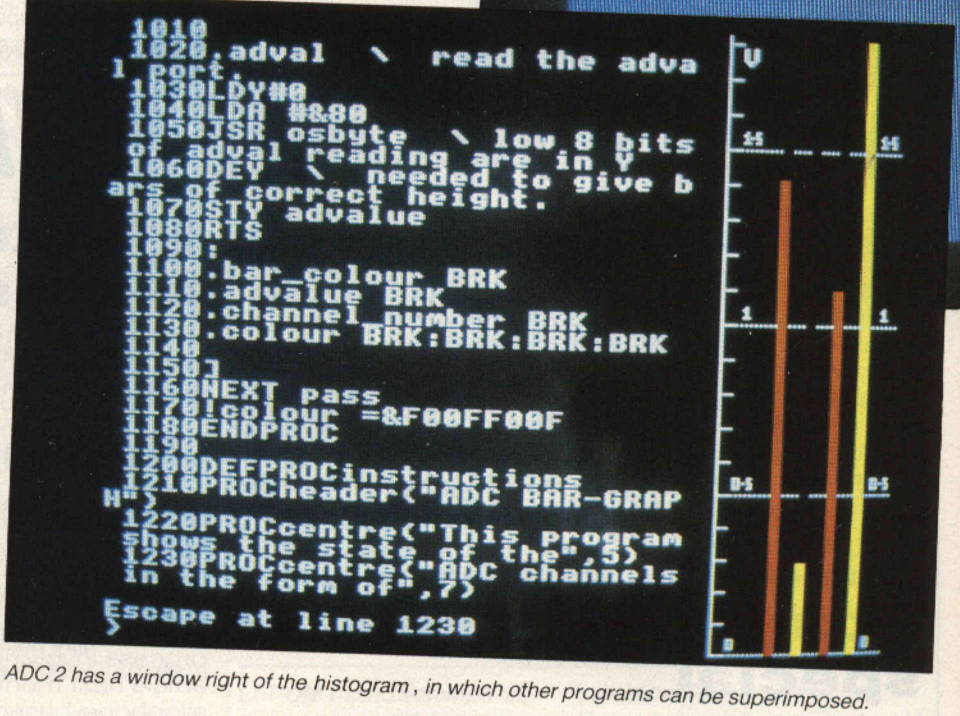
get adval: does an osbyte call to obtain the current voltage at one of the channels of the ADC port. X contains the number of the channel (1 to 4) and after the call the most significant 8 bits of the result is in the Y register. Since the vertical bar is 256 units high we only need these 8 bits. Without averaging, that is just about the accuracy of the AD converting chip anyway.

draw bars: having got the port value and the colour for the bar, this routine does the bar drawing. First, it works out where to start plotting the bar. If you want to shift the bars over, or indeed to use the routine for a fancy multi-bar spectrum analyser, you must modify this part of the routine. Plot loop does the actual plotting; it starts plotting at the top of the screen, and as the bar probably hasn't started yet, it plots empty space. Once the bar is reached (line 820) the bar colour is loaded into the accumulator. Every time eight 'rungs' of the bar have been done, another &280 has to be added to the plotting address to get to the next screen line. This is the number of memory bytes taken by each of the 32 character lines.



```
ADC

Press
any
key
to
change
channel
```



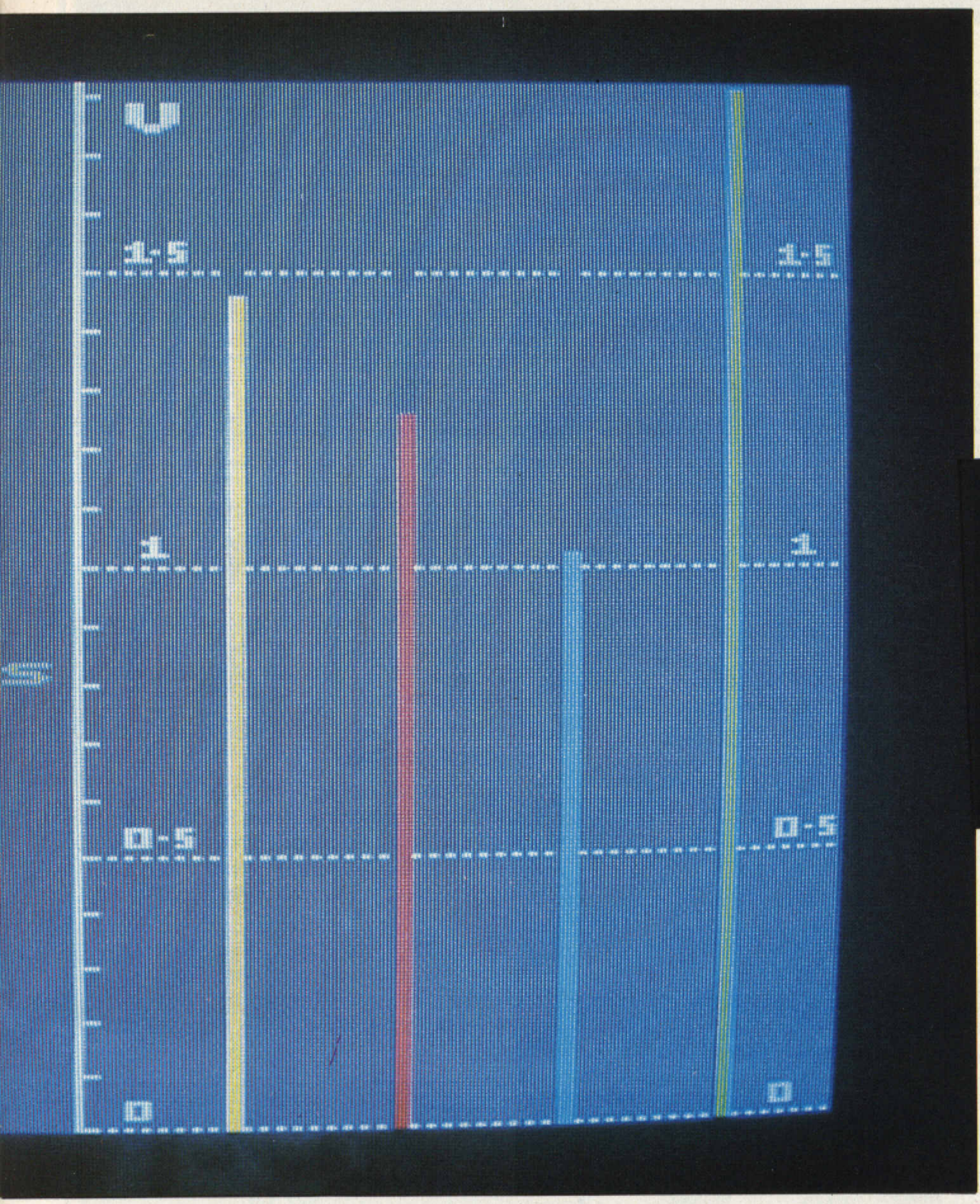
```
 1010
 1020.adval   \   read the adva
l port.
 1030LDY#0
 1040LDA #&80
 1050JSR osbyte  \ low 8 bits
 of adval reading are in Y
 1060DEY  \  needed to give b
ars of correct height.
 1070STY advalue
 1080RTS
 1090:
 1100.bar_colour BRK
 1110.advalue BRK
 1120.channel_number BRK
 1130.colour BRK:BRK:BRK:BRK
 1140
 1150]
 1160NEXT pass
 1170!colour =&F00FF00F
 1180ENDPROC
 1190
 1200DEFPROCinstructions
 1210PROCheader("ADC BAR-GRAP
H")
 1220PROCcentre("This program
 shows the state of the",5)
 1230PROCcentre("ADC channels
 in the form of",7)
Escape at line 1230
>
```

ADC 2 has a window right of the histogram, in which other programs can be superimposed.

Finally, the whole process is repeated for each of the channels.

The rest of the program is in BASIC. PROCscreen draws the axis and puts on the graduations and calibrations. Because MODE 2 has such chunky numbers I got my son to produce the VDU23s for some neater ones (thanks Nathan). The procedure also finds out how many channels you wish to observe. Unconnected channels tend to bounce around somewhat distractingly so it is a good idea not to plot them. But in addition, by only enabling the channels you wish to observe (lines 1300 – 1330) the time between successive port readings is decreased.

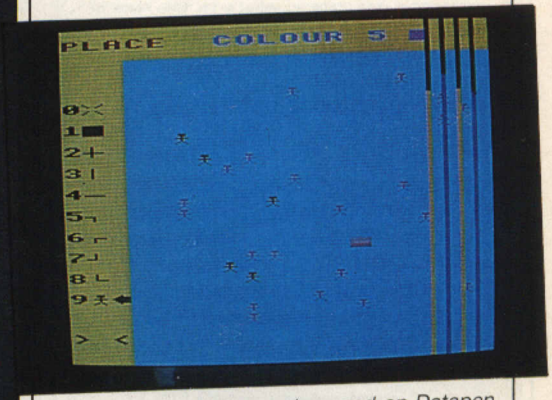
To test the program out, the simplest thing to do is to attach joysticks to the port. Twiddling the sticks should make the bars change height smoothly. Rotate the stick to observe two bars performing simple har-



ADC 1 is a full-screen histogram representation of voltages at each of the four ADC inputs.

monic motion 180 degrees out of phase (that should please Physics teachers like myself). It is a simple matter to make an interface board with 4mm sockets so that you can access the individual channels. Then you can start monitoring light levels with circuit A **(Figure 1)**, temperatures with circuit B **(Figure 2)**, even weights with circuit C **(Figure 3)**, and so on. If you come up with any neat ideas *Electronics and Computing* would be interested to hear of them.

Just a word about calibration. The scale graduations were in just about the correct position for my machine, but there might be variations. The STEP 56 in line 1460 determines the position of the graduations.



The bars of ADC 2 superimposed on Datapen lightpen software.

If you change the step value, you should also modify the step in line 1500 to be 5 times the step in line 1460.

The second program, given in **Listing 2,** is based on the same bar plotting routines as ADCbars. But in this program the bars go on being plotted, and hence monitoring the analogue port, even after the program finishes. This is done by using the 'event' facility of the BBC which I described in my earlier Simuled article. The event used is the ADC conversion completed event. Each time this occurs, the ADCbars routine

LISTING 1 (ADC 1)

```
100 REM/// A/D BAR-DISPLAY ///
110 MODE7
120 *TV0,1
130 PROCassemble
140 PROCinstructions
150 MODE2
160 PROCscreen .
170 :
180 REPEAT
190   CALL adc_bars      .
200   UNTIL NOT INKEY-129 : REM any key has been pressed.
210 RUN
220 :
230 DEF PROCscreen
240 VDU19;5;0;0;0; : REM colour background.
250 COLOUR 2 : REM  letters in green.
260 PRINT"  ADC"
270 PRINT"  ==="
280 PRINTTAB(1,12)"Press"'"  any"'"  key"'"  to"'" change"'"channels"
290 PROCaxis
300 VDU23;8202;0;0;0; :REM remove cursor.
310 ENDPROC
320 :
330
340 DEF PROCassemble
350 code=&900 : REM you could DIMcode 120 if you prefer.
360 osbyte=&FFF4
370 screen=&80
380
390 FOR pass =0 TO 2 STEP 2
400   P%=code
410   [OPT pass
420   :
430   .adc_bars
440   LDA channel_number
450   PHA                    \ save the number of channels on the stack.
460   .main_loop
470   JSR get_adval
480   JSR draw_bars
490   DEC channel_number
500   BNE main_loop
510   PLA
520   STA channel_number \ replace the number of channels.
530   RTS
540
550   .get_adval
560   LDX channel_number
570   LDA colour-1,X  \ Get channel colour.
580   STA bar_colour  \ and store it.
590   JSR adval
600   RTS
610
620   .draw_bars
630   LDX #&FF  \  X is the plotting index up the bar.
640   LDY channel_number
650   CLC
660   LDA #0
670   .adjust  \ the value of (screen) for each bar.
680   ADC #80  \ gives the space between bars.
690   DEY
700   BNE adjust
710   STA screen
720   LDA #&31
730   ADC #0
740   STA screen+1
750   LDA #0  \  start the bar with no colour.
760   :
770   .plot_loop
780   STA (screen),Y
790   DEX
800   CPX advalue     \ time for the coloured bar?
810   BNE go_down
820   LDA bar_colour \ yes it is.
830   .go_down
840   INY
850   CPY #8  \  is it time to add &280 to screen address?
860   BNE plot_loop
870   :
880   PHA      \ save the bar colour.
890   LDY #0
900   LDA screen
910   CLC
920   ADC #&80
930   STA screen
940   LDA screen+1
950   ADC #2
```

LISTING 1 Cont.

```
 960   STA screen+1   \ &280 has been added
 970   PLA            \ get the bar colour back
 980   CPX #&FF       \ bar finished yet?
 990   BNE plot_loop
1000   RTS
1010
1020   .adval  \  read the adval port.
1030   LDY#0
1040   LDA #&80
1050   JSR osbyte  \ low 8 bits of adval reading are in Y
1060   DEY  \  needed to give bars of correct height.
1070   STY advalue
1080   RTS
1090   :
1100   .bar_colour BRK
1110   .advalue BRK
1120   .channel_number BRK
1130   .colour BRK:BRK:BRK:BRK
1140
1150   ]
1160   NEXT pass
1170 !colour =&0C0C0C0F : REM colour values for the 4 bars.
1180 ENDPROC
1190
1200 DEFPROCinstructions
1210 PROCheader("ADC BAR-GRAPH")
1220 PROCcentre("This orogram shows the state of the",5)
1230 PROCcentre("ADC channels in the form of",7)
1240 PROCcentre("vertical bars.",9)
1250 PROCcentre("How many channels",14)
1260 PROCcentre("do you wish to observe? ",16)
1270 REPEAT:J%=GET-48:UNTIL J%>0 AND J%<5
1280 num_channels=J%
1290 ?channel_number=num_channels :REM store for use of the m. code.
1300 IF num_channels=1 THEN *FX16.1
1310 IF num_channels=2 THEN *FX16,2
1320 IF num_channels=3 THEN *FX16,3
1330 IF num_channels=4 THEN *FX16,4
1340 ENDPROC
1350 :
1360 DEFPROCcentre(A$,vpos) PRINTTAB((19-LENA$DIV2),vpos);A$;:ENDPROC
1370 DEFPROCheader(A$) VDU30:FORI=0TO1:PRINTCHR$&81;CHR$&9D:CHR$&83;
CHR$&8D::PROCcentre(A$,VPOS):PRINT:NEXT:ENDPROC
1380 :
1390 DEFPROCaxis
1400 VDU 23.230,0,224,160,160,160,224,0,0:REM 0
1410 VDU 23,231,0,227,162,171,161.227,0,0:REM 0.5
1420 VDU 23,232,0,16,48,16,16,56,0,0:REM 1
1430 VDU 23,233,0,70,196,86,66,230,0,0:REM 1.5
1440 GCOL 0,7 :REM set colour to white.
1450 MOVE 530,0:DRAW 530,1080
1460 FOR J%=0 TO 1100 STEP 56
1470   MOVE 530,J%:DRAW 550,J%                 :REM graduations.
1480   NEXT
1490 VDU5
1500 FOR J%=0 TD 1100 STEP 280
1510   MOVE 530,J%:PLOT 21,1279,J%             :REM dotted line.
1520   PRINTCHR$11;CHR$8;CHR$(230+J%/270):REM numbers on right.
1530   MOVE 580,J%:
1540   PRINTCHR$11;CHR$(230+J%/270)            :REM numbers on left.
1550   NEXT
1560 MOVE 570,1000:PRINT"V"  1570 ENDPROC
```

is called which updates the bars on the screen.

The listing is fully documented with REMs and assembler comments. In order to leave more space for your own program to use once the bars are being produced, Mode 1 is used. The appearance is similar to that of the first program but is shifted to the right. Two colours, red and yellow, are used for the bars. Once the program has

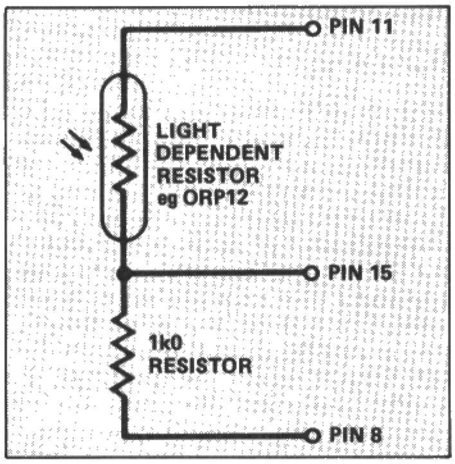


Figure 1. Light sensor, circuit A.

been run, a text window is created leaving the bars being plotted on the right. You can now run a new program in that space provided it stays in Mode 1 and doesn't use any of page &900 or zero page locations &B0 and &B1.

You may care to add one or two improvements to the program. For example, it does not check for Mode as was done in Simuleds. The same method could be used. And it is probably possible to find out which was the last channel to be read and hence to update it only. This might result in slightly less time being taken up by the routine.

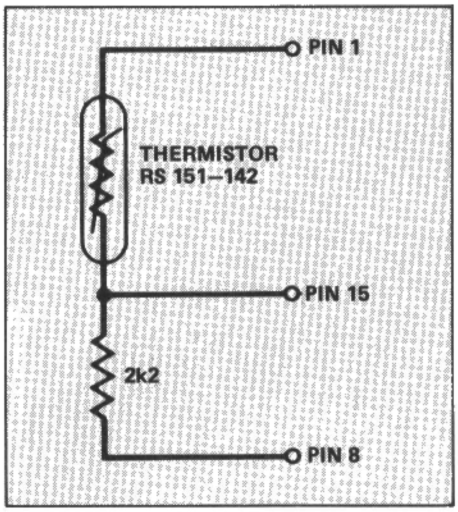


Figure 2. Heat sensor, circuit B.

In fact, the penalty you pay for having continuous port monitoring is a loss in BASIC speed. Try this test. In immediate mode type

TIME=0:FOR L=1 TO 1000:
NEXT L:P.TIME

This usually gives the result 64 on my machine, but with the bars routine running it slows to somewhere around 120. The event can be switched of with *FX13,3 to regain speed, then switched back on with *FX14,3.

The author can supply a cassette or 40-track disc of these programs, by sending cheques or postal orders for £5 to: *M. E. Williams, 11 Cressy Road, London NW3 2NB.*

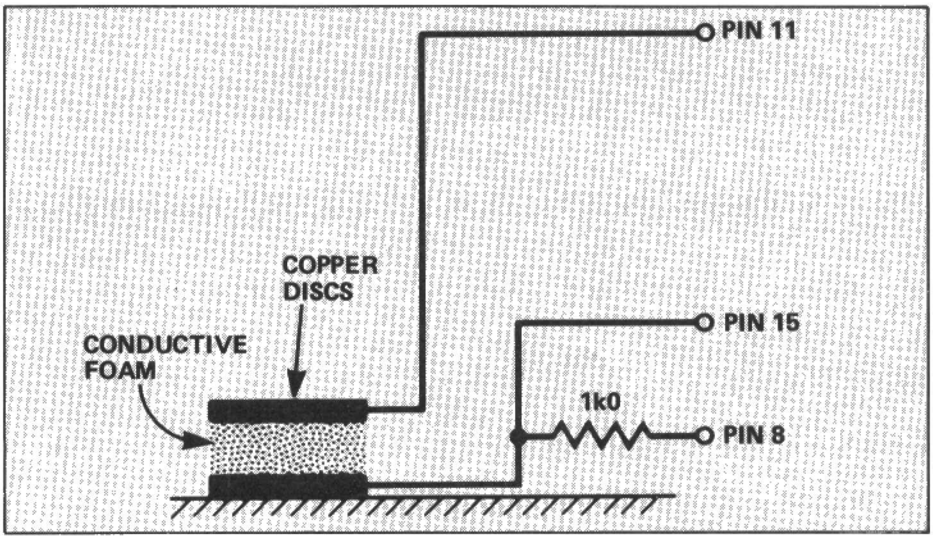


Figure 3. Conductive foam scales, circuit C.

LISTING 2 (ADC 2)

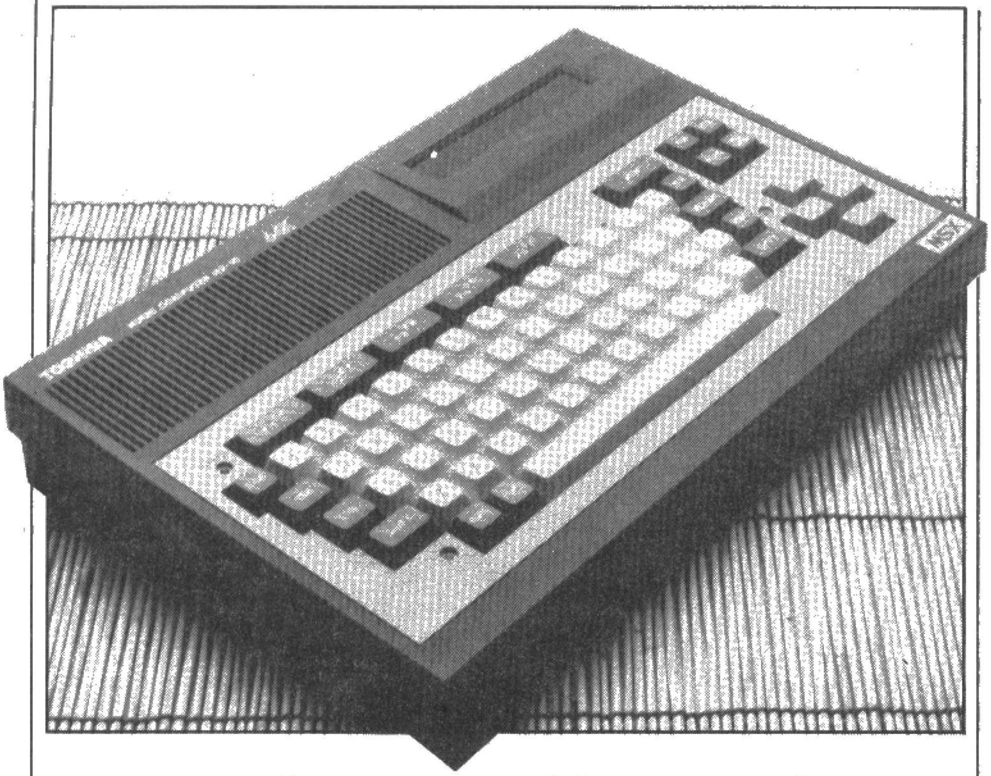
```
100 REM/// EVENT DRIVEN A/D BAR-DISPLAY ///
110 REM First disable the ADC event.
120 *FX13.3
130 MODE7
140 *TV0,1
150 PROCassemble
160 PROCinstructions
170 MODE1
180 PROCscreen
190 CALL code
200 END
210 :
220 :
230 DEF PROCscreen
240 PROCaxis
250 VDU28.0.31.28,0 :REM text window
260 ENDPROC
270 :
280
290 DEF PRDCassemble
300 code=&900 : REM you could DIMcode 140 if you prefer.
310 osbyte=&FFF4
320 screen=&80
330
340 FOR pass =0 TO 2 STEP 2
350   P%=code
360   [OPT pass
370   :
380   LDA #(code+18) MOD 256:STA &220    \ set the event vectors
390   LDA #(code+18) DIV 256:STA &221
400   LDA #14:LDX #3:JSR &FFF4    \ enable ADC conversion event.
410   RTS
420   .adc_bars
```

LISTING 2 Cont.

```
 430   TXA:PHA              \ save X register during the event.
 440   LDA channel_number
 450   PHA                  \ save the number of channels on the stack.
 460   .main_loop
 470   JSR get_adval
 480   JSR draw_bars
 490   DEC channel_number
 500   BNE main_loop
 510   PLA
 520   STA channel_number \ replace the number of channels.
 530   PLA:TAX              \ restore the X register after the Event.
 540   RTS
 550
 560   .get_adval
 570   LDX channel_number
 580   LDA colour-1,X   \ Get channel colour.
 590   STA bar_colour   \ and store it.
 600   JSR adval
 610   RTS
 620
 630   .draw_bars
 640   LDX #&FF   \  X is the plotting index up the bar.
 650   LDY channel_number
 660   CLC
 670   LDA #&0
 680   .adjust  \ the value of (screen) for each bar.
 690   ADC #24  \ gives the space between bars.
 700   DEY
 710   BNE adjust
 720   STA screen
 730   LDA #&32
 740   STA screen+1
 750   LDA #0  \  start the bar with no colour.
 760   :
 770   .plot_loop
 780   STA (screen),Y
 790   DEX
 800   CPX advalue      \ time for the coloured bar?
 810   BNE go_down
 820   LDA bar_colour \ yes it is.
 830   .go_down
 840   INY
 850   CPY #8  \  is it time to add &280 to screen address?
 860   BNE plot_loop
 870   :
 880   PHA      \ save the bar colour.
 890   LDY #0
 900   LDA screen
 910   CLC
 920   ADC #&80
 930   STA screen
 940   LDA screen+1
 950   ADC #2
 960   STA screen+1  \ &280 has been added
 970   PLA           \ get the bar colour back
 980   CPX #&FF      \ bar finished yet?
 990   BNE plot_loop
1000   RTS
1010
```

```
 1020   .adval  \  read the adval port.
 1030   LDY#0
 1040   LDA #&80
 1050   JSR osbyte  \ low 8.bits of adval reading are in Y
 1060   DEY  \  needed to give bars of correct height.
 1070   STY advalue
 1080   RTS
 1090   :
 1100   .bar_colour BRK
 1110   .advalue BRK
 1120   .channel_number BRK
 1130   .colour BRK:BRK:BRK:BRK
 1140
 1150   ]
 1160   NEXT pass
 1170 !colour =&F00FF00F
 1180 ENDPROC
 1190
 1200 DEFPROCinstructions
 1210 PROCheader("ADC BAR-GRAPH")
 1220 PROCcentre("This program shows the state of the",5)
 1230 PROCcentre("ADC channels in the form of",7)
 1240 PROCcentre("vertical bars.",9)
 1250 PROCcentre("How many channels",14)
 1260 PROCcentre("do you wish to observe? ",16)
 1270 REPEAT:J%=GET-48:UNTIL J%>0 AND J%<5
 1280 num_channels=J%
 1290 ?channel_number=num_channels :REM store for use of the m. code.
 1300 IF num_channels=1 THEN *FX16,1
 1310 IF num_channels=2 THEN *FX16,2
 1320 IF num_channels=3 THEN *FX16,3
 1330 IF num_channels=4 THEN *FX16,4
 1340 ENDPROC
 1350 :
 1360 DEFPROCcentre(A$,vpos)PRINTTAB((19-LENA$DIV2),vpos);A$;:ENDPROC
 1370 DEFPROCheader(A$)VDU30:FORI=0TO1:PRINTCHR$&81;CHR$&9D;CHR$&83;
CHR$&8D;:PROCcentre(A$,VPOS);PRINT:NEXT:ENDPROC
 1380 :
 1390 DEFPROCaxis
 1400 axis=980
 1410 VDU 23,230,0,224,160,160,160,224,0,0:REM 0
 1420 VDU 23,231,0,227,162,171,161,227,0,0:REM 0.5
 1430 VDU 23,232,0,16,48,16,16,56,0,0:REM 1
 1440 VDU 23,233,0,70,196,86,66,230,0,0:REM 1.5
 1450 GCOL 0,7 :REM set colour to white.
 1460 MOVE axis,0:DRAW axis,1080
 1470 FOR J%=0 TO 1100 STEP 56
 1480   MOVE axis,J%:DRAW axis+20,J%           :REM graduations.
 1490   NEXT
 1500 VDU5
 1510 FOR J%=0 TO 1100 STEP 280
 1520   MOVE axis,J%:PLOT 21,1279,J%
 1530   PRINTCHR$11;CHR$8;CHR$(230+J%/270):REM numbers on right.
 1540   MOVE axis+30,J%
 1550   PRINTCHR$11:CHR$(230+J%/270)      :REM numbers on left.
 1560   NEXT
 1570 MOVE axis+20,1000:PRINT"V"
 1580 VDU4
 1590 ENDPROC
```

MSX – the soft options



MSX BASIC features a wide range of commands. Gary Evans examines the language's specification.

There will be some fun and games later this year when the magazines that make their living from reviewing and comparing various microcomputers contend with the plethora of MSX computers that will soon be in the shops. The reason for predicting more than a few problems for these magazines is, of course, that in essence **all** MSX computers will be the same. More than one computing title will probably find itself resorting to comments on the colour of a particular machine's keyboard and discussions about the length of mains cable supplied with the various designs.

Needless to say *E&CM* will not be indulging in such an exercise. We have managed to obtain a copy of the MSX specification in the form in which it is supplied to manufacturers joining under the MSX umbrella organisation in the UK. (Obviously we were not privileged to an assembly listing of the MSX OS and BASIC for fear that we might give too much away and cause an outbreak of pirate MSX micros). Our approach to the MSX story will be to give general details of the standard which will apply equally to **any** machine bearing the MSX badge. In some cases, where reference to a particular machine is made this will be the Toshiba HX 10 which is likely to be the first computer to reach these shores – the scheduled launch date being mid September.

Our report last month gave an overview of the hardware of a typical MSX computer (or even of MAX computers as one of our typos had them). This month we'll be concentrating on the software aspects of the specification.

Before moving on from the hardware though there is just one, fairly trivial point that we'll mention for the sake of completeness. This concerns the Japanese flair for production design. In view of the fairly shoddy lumps of plastic that provide houses for some British micros it is refreshing to note that, without exception, all of the MSX machines that we have seen to date have been built to stand some fairly rough handling. All have sturdy enclosures complemented by keyboards built to a high specification. All of the micros have built in power supplies – a much more satisfactory approach than the combined transformer/plug contraptions that seem to be favoured in home-grown micros. In the case of the Toshiba HX 10 this did lead to a few heat dissipation problems with the computer running warm, if not hot. But this did not lead to any malfunction of the computer even after an all night soak test.

'we have managed to obtain a copy of the full MSX specification'

All singing

In view of the markets at which MSX micros are aimed, the up market games/home computer, then their competence or otherwise in the areas of graphics and sound will be of great importance.

The command PLAY may be followed by between one and three string expressions to control the output of the three sound channels supported by the MSX hardware. The specification states, somewhat optimistically that the PLAY command embeds a "micro music language" into a character string. The string in general consists of single character commands ending in a null string to terminate any sequence. There are a number of commands associated with PLAY, the major one taking the form of an alpha character in the range A to G together with an optional #, + or – symbol. This command plays a note in the currently selected octave with the option of sharpening it with the + or # suffix or of flattening it with the addition of the – sign. For any of you who know which notes correspond to the black keys on a piano the warning that B# is not a valid note indicates that you can only sharpen those notes that can be so treated on a piano.

The O command can select one of 8 octaves each ranging from C to B while a Nn entry in the music string will play note n from the full 96 note range (this for those of you still with us is simply another way of specifying a note without bothering with the O command).

L will set the length of notes and can range from 1 (a whole note) to 164 (yes, a 64th of a note). A rest can be incorporated in a string by means of the R call, and its associated parameter and the PLAY command supports 'dotted notes'; a dot after a note or indeed pause will scale its length by 3/2.

Tempo, modulation and volume are set by the letters T, M and V respectively while an S command will invoke one of 8 different envelopes when followed by the appropriate parameter.

If all of the above were not enough to satisfy most peoples needs MSX BASIC also supports a SOUND command that allows values to be written directly to the

PSG's registers.

. . . All dancing

The major graphics control is provided by the SCREEN statement which has the following form:

SCREEN [<mode>] [,<sprite size>]
[,<key click switch>] [,<cassette
baud rate>] [,<printer option>]

As can be seen this command deals with quite a number of the important attributes of the video display plus other aspects of the system's configuration.

The mode parameter selects from one of four modes these being:

0: 40 x 24 text
1: 32 x 24 text
2: high resolution
3: multi colour

Sprite size selects the basic way in which the sprite graphics definition is handled; the modes are:

0: 8x8 unmagnified
1: 8x8 magnified
2: 16x16 unmagnified
3: 16x16 magnified

A full description of sprite graphics is beyond the scope of this article but for those of you who wish for a detailed explanation of this concept we would suggest that you get hold of the Texas instruments data sheet for the TMS9918 VDP that forms the heart of the MSX graphics circuitry.

The remaining parameters of the SCREEN command are reasonably self explanatory. The full listing of the commands supported by MSX reveals that there are many more commands dealing with the graphics display of MSX standard computers. Many of these will be familiar and, again, self explanatory, although you will notice that the commands INK and PAPER are missing from the list. Instead the COLOUR command can have up to three parameters associated with it, the syntax taking the form:

COLOUR[<foreground colour>]
[,<background colour>]
[,<border colour>]

The DRAW command is once again described rather grandly as a graphic micro language by the MSX user's Bible but to be fair does include a healthy array of commands. These include U,D,L,R together with associated parameters that will move the graphics 'pen' in the four (obvious) directions together with a set of four commands covering diagonal movement. Commands covering orientation, colour and scaling factors are also amongst other parameters associated with the DRAW command.

A final point before leaving graphics is to point out the ON SPRITE GOSUB and SPRITE ON/OFF/STOP which are commands associated with the detection of sprite collisions will prove invaluable in a number of different situations.

Odds and ends

There is far more to MSX BASIC than the sound and graphics functions so far described, as a brief look at the specification will reveal. Unfortunately there is little to please the structured programmer, no BBC-like PROC statements but there are other welcome additions to the language such as the ON ERROR GOTO command. One minor irritation is the rather terse error messages generated by this implementation of BASIC. A few years ago when bits cost money there was an excuse for this but today with memory so cheap and with 32K of ROM to play with there seems no reason for not building in far more communicative error messages.

EOF

MSX BASIC should prove a popular implementation of the language capable of satisfying both the tyro programmer and the more advanced user who will be able to make use of the many bells and whistles built into the standard. The fact that there will be a standard means that once the BASIC on a particular machine is mastered any number of MSX micros will be immediately familiar.

Next month we will look beyond the bounds of BASIC with an in-depth look at the configuration of a typical system including memory maps and a detailed description of the cartridge port that forms the mains expansion route for the computers.

Complete list of MSX commands and statements.

AUTO
CONT
DELETE
LIST
LLIST
NEW
RENUM
RUN
TRON/TROFF
CLEAR
DATA
DIM
DEFINT
DEFSNG
DEFDBL
DEFSTR
DEF FN
DEFUSR
ERASE
END
ERROR
FOR
NEXT
GOSUB
RETURN
GOTO
IF THEN
IF GOTO
INPUT
LINE INPUT
LET
MIDS

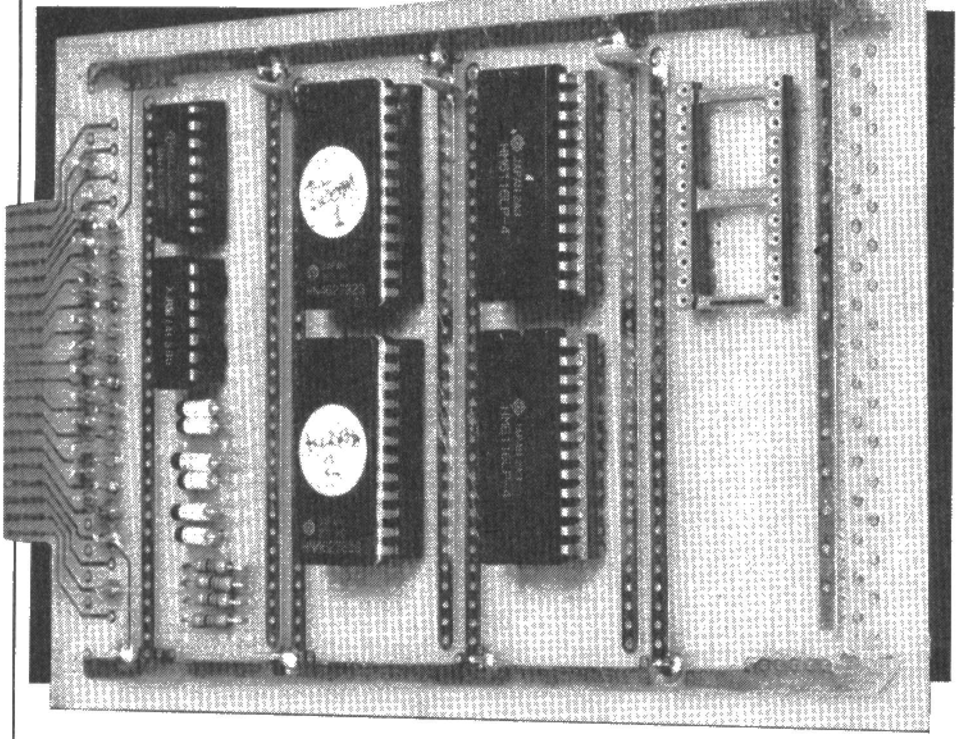
ON ERROR GOTO
ON
ON
POKE
PRINT
PRINT USING
READ
REM
RESTORE
RESUME
RESUMEO
RESUME NEXT
RESUME (line number)
STOP
SWAP
ABS(X)
ASC(XS)
ATN(X)
BINS(n)
CDBL(X)
CHRS(I)
CINT(X)
COS(X)
CSNG(X)
CSRLIN
ERL/ERR
EXP(X)
FIX(X)
FRE(O)
FRE("")
HEXS(X)
INKEYS

INPUTS(X)
INSTR(I,)XS,YS)
INT(X)
LEFTS(XS,I)
FLEFTS(XS,I)
LEN(XS)
LOG(X)
LPOS(X)
MIDS(XS,I(,J)
OCTS(n)
PEEK(I)
POS(I)
RIGHTS(XS,I)
RND(X)
SGN(X)
SIN(X)
SPACES(X)
SPC(I)
SQR(X)
STRS(X)
STRINGS(I,J)
STRINGS(I,XS)
TAB(I)
TAN(X)
USR(digit)(X)
VAL(XS)
VARPTR (variable name)
VARPTR (file number)
SCREEN
WIDTH
CLS
LOCATE

COLOR
PUT SPRITE
CIRCLE
DRAW
LINE
PAINT
PSET
PRESET
KEY
KEY LIST
KEY ON/OFF
ON KEY GOSUB
KEY (function key)
ON/OFF STOP
ON STRIG GOSUB
STRIG
ON STOP GOSUB
STOPE ON/OFF/STOP
ON SPRIRITE GOSUB
(line number)
SPRITE ON/OFF/STOP
ON INTERVAL
INTERVAL ON/OFF/STOP
VPOKE
BEEP
MOTOR
SOUND
PLAY
On
Nn
Ln
Rn

Tn
Vn
Mn
Sn
OPEN
PRINT
INPUT (file n,o)
LINE INPUT
INPUTS
CLOSE
SAVE
LOAD
MERGE
BSAVE
BLOAD
BSAVE
CLOAD
CLOAD?
CALL
POINT
VPEEK
STICK
STRING
PDL
PAD
PLAY
EOF
TIME
SPRITES
VDP
BASE
OUT

WAIT
INP
AUTO
CONT
DELETE
LIST
MAX FILES
LPRINT
LPRINT USING



Dragon proto board

Huw Jones describes a versatile prototyping system for the development of EPROM based software.

When used with one of the 6809 assembly language packages around, the Dragon/EPROM programmer combination forms a very reasonable microprocessor software development aid.

M J Kerry's "Dream" and "Dreambug" package for Dragon Data offers excellent value for money. These may be purchased as two separate items which are then combined into a single 8K position – independent utility, "Alldream". This suffers from the handicap of residing in RAM which is more often than not corrupted during the early stages of debugging an errant object code program. Dragon data have now remedied this situation by issuing a "Dream" cartridge. However, for those who have already bought the cassette versions then the EPROM programmer offers an ideal opportunity to create your own "Alldream" development card. This is an approach which is encouraged by the "Dreambug" manual itself.

A schematic for a simple firmware development board is given in **Figure 1.** This can be very quickly constructed on another prototyping card which is shown in the photo. It contains two 2732s for "Alldream" which resides at $C000-$DFFF. Execution addresses now become &HC080 and &HD404 respectively for the two constituent parts. Another socket is provided, PROM 3, for convenient debug access to prototype software in a 2732. In addition 4K of static RAM, in the form of two 6116s, can be added at $F000-$FEFF.

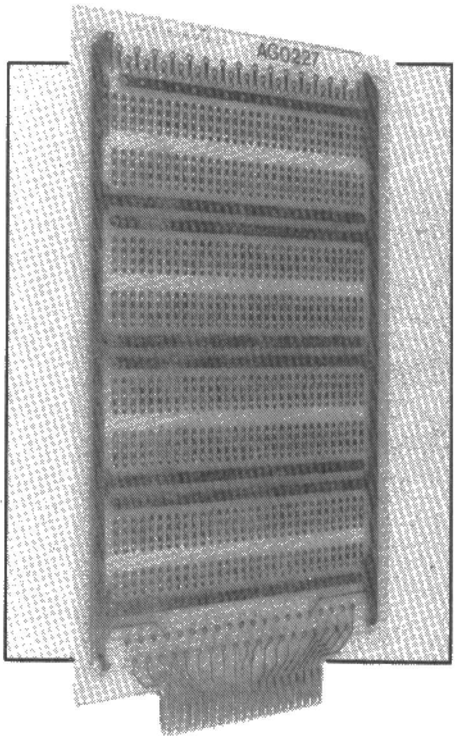
The diodes shown are the preferred Ge OA47 type. These were chosen for low forward voltage drop. However, if you can't easily lay your hands on one then the 1N4148 will work most satisfactorily in virtually all instances.

The card described here offers a number of advantages over Dragon's own cartridge version.

Dragon data

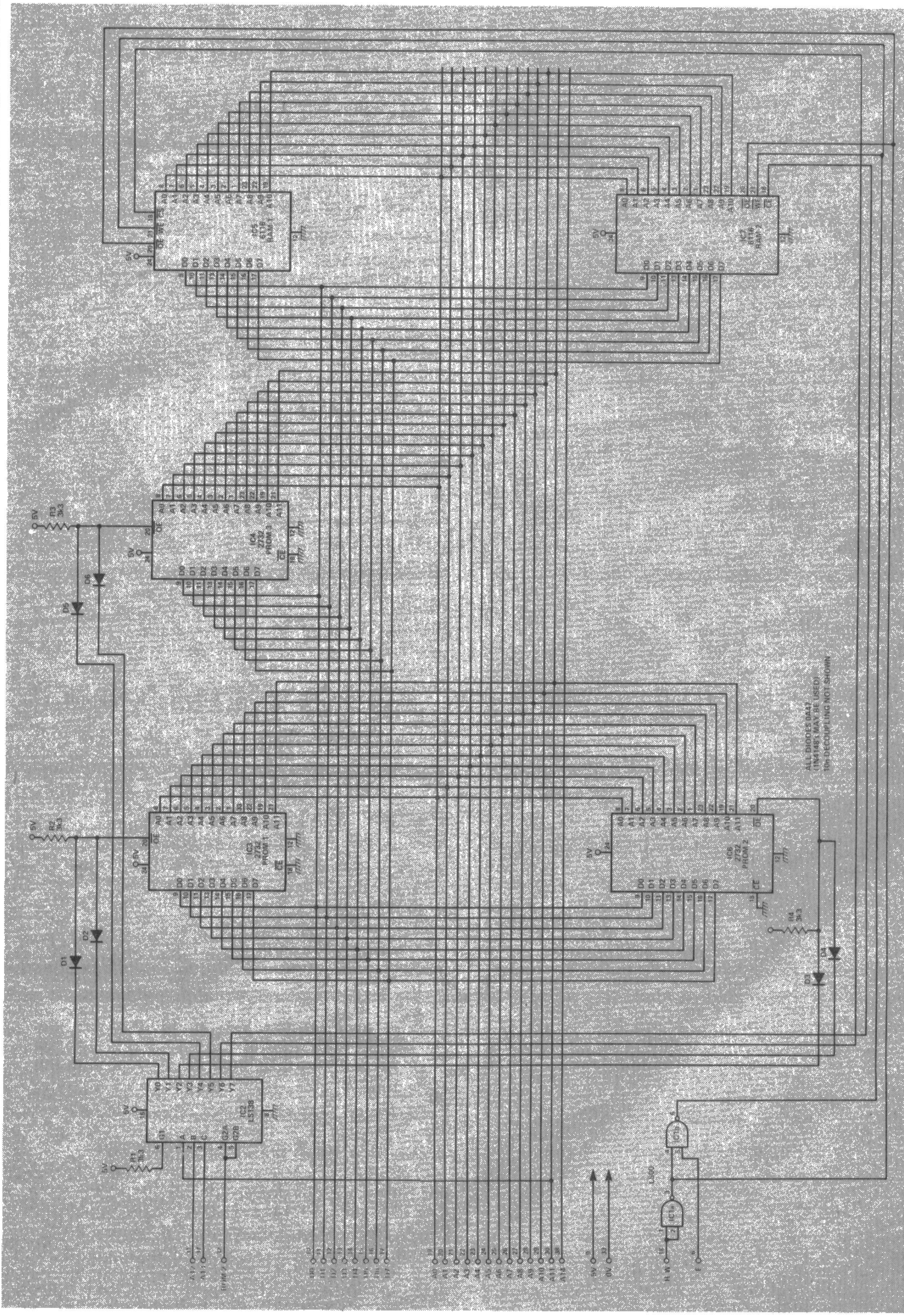
If there is sufficient demand from users we will consider designing a PCB board for the development system but the prototyping board approach has a number of advantages. First and foremost is the fact that it is in the nature of development work that changes to a circuit will have to made during hardware/software debugging and the method of construction chosen readily accommodates this. Secondly, the board provides space, albeit a small amount, in which additional hardware may be mounted.

As a development system to be used in conjunction with the "Alldream" utility, the card described here offers a number of advantages over Dragon's own cartridge version of the software. The ready access to the 4K RAM and to a third EPROM that can house development software is something that cannot be matched by the official cartridge based software without resorting to some form of expander for the cartridge slot; the prices charged for some of these will approach the total cost of our board.



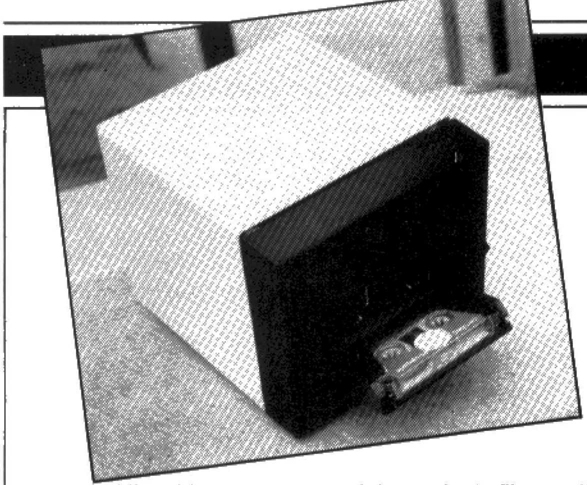
The bare prototyping board.

Steve's Electronics, Castle Arcade, Cardiff have arranged to supply all the components for the software development board and the Dragon EPROM programmer board described in the past two issues.



ALL DIODES OA47 (1N4148's MAY BE USED) 10n DECOUPLING NOT SHOWN

Figure 2. Simple development board.



ULTRADRIVE

Would you be loopy to buy an Ultradrive? William Owen decides no – well not necessarily

The Ikon Ultradrive poses one sticky problem for BBC owners considering the best means of data storage. The argument between disk drives and cassette recorders is a simple one – price against performance – but with the Ultradrive we have a new dimension; a classic compromise between the two.

Ultradrive is a looped microcassette tape drive with a controller and software. The first time buyer gets the main unit, power cable, ribbon cable, ROM and a utilities cassette.

The drive is attached to the BBC via the user port and the power output, and the ROM inserted into the sideways ROM board and accessed with the command *ULTRA. A simple operation if you haven't already got an overflowing ROM board (if, like me, you have, then something has to go).

The first task is to format a blank microcassette using the utilities provided. The write enable plug is removed from the cassette and with the old *CAT a list of utilities will be displayed within 5 seconds.

We are interested in FORM, and once *FORM TEST is entered a message MOUNT TAPE Y/N appears on the screen. The blank cassette is then inserted in the Ultradrive; type Y, and a rather lengthy process of formatting takes place. The manual says it will take 'about 3 minutes' – it took 5, but this is the only time when using Ultradrive that you will have to wait longer than a minute for anything.

Formatting splits the tape into two 'loops', each independent of the other and with its own catalogue. *CAT gives information about the programs on the tape and the remaining storage space. The catalogue is rewritten automatically each time a new file is saved.

Storage available

Under normal conditions each loop will store about 35K of data. If files are required to be longer than this, the number of blocks (up to 255) in the first loop can be specified at the time of formatting, allowing a maximum file length of just under 64K.

Programs and data files are stored in continuous blocks on the tape. The length of each block is 256 bytes. The system will open a new file or save a new program at the start of the largest available gap on the tape. In the case of a program, the file produced is normally just large enough to contain it, whereas with a data file the system will reserve 20 adjacent blocks (5K) for the file, or the largest gap available. In both cases a number of blocks can be specified at the time of opening the file.

The use of adjacent blocks allows random access files to be used relatively speedily. The storage system can calculate which block any particular piece of data is in and go straight to it. The full range of random access commands for the BBC are supported by this system.

A more complex system can be initialised, which allows two drives and up to 5 files to be in use at the same time. PAGE is moved up from the normal E00 to 1600, giving facilities roughly equivalent to a disk system.

Utilities

The various utilities are more difficult to assess because of the inadequacy of the manual. This is, according to Ikon, a provisional document (7 photocopied sheets of A4 paper) and 'not very user friendly'. It isn't, and until the properly printed and edited manual arrives many facets of the utilities will remain a mystery.

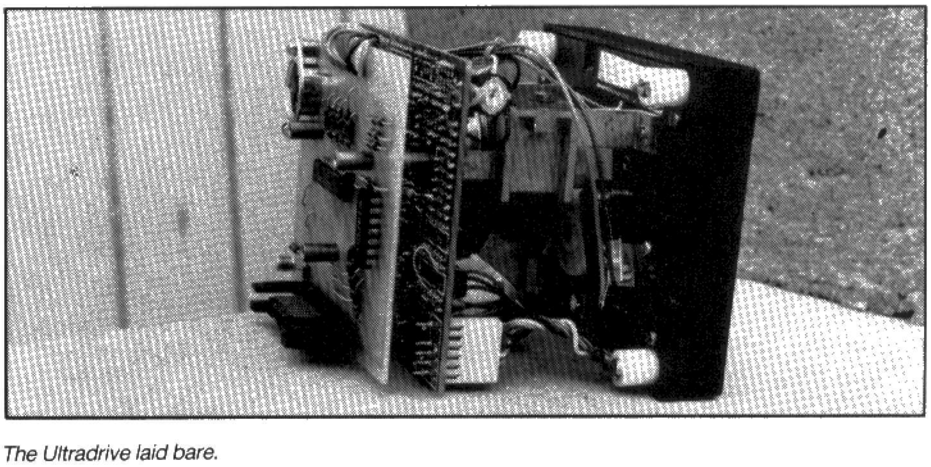
The seven utilities are ACCESS, INFO, RENAME, DELETE, FORM, RECOUP, and UTICPY. there is no information on INFO, RECOUP or UTICPY; the purpose of DELETE, FORM(at) and RENAME are obvious. ACCESS is used to lock or unlock files to protect them against unauthorised use or accidental overwriting. Tapes can also be write-protected by removing the plastic studs from the corner of the cassette.

In terms of performance and speed, the Ultradrive is obviously inferior to a disk system. However with certain exceptions, most users will have no time to become impatient. Cataloguing a loop takes between 3 and 10 seconds. Loading a program takes between 2 and 20 seconds, depending on length and the position on the tape. Saving is reasonably quick: it took 15 seconds to load a 160 line BASIC program onto a clean loop. The biggest problem is in swopping loops. The user does not know which loop he will access first, except that it will be that which he last used. Swopping from one loop to another (using *SWAP) takes between 30 and 50 seconds which is generally unsatisfactory (cassette recorder users may think differently).

As a tool for the amateur programmer the Ultradrive would be a low cost and efficient alternative to disk drives. the system is far superior to cassette recorders which retail at about a half to one third of the price, and in particular, potential buyers (anyone who can't afford a disk drive) should bear in mind the immensely useful cataloguing and access facilities. Nevertheless there are important drawbacks, the most obvious being lack of software support. There are very few games and utilities written to microcassette; these could be only be loaded from cassette to Ultradrive if they were not copy protected, and anyway both storage systems would have to be available.

DATAFILE

Product	Ultradrive
Supporting	The BBC micro
Price	£79.95 plus £3.45 p&p
Supplier	Ikon Computer Products Kiln Lake Laugharne Dyfed SA33 4QE



The Ultradrive laid bare.

Counting micro seconds

Paul Beverley, author of the new BBC micro service manual, examines the use of the 16-bit counter/timers of the 6522 VIA.

Since computers can execute hundreds of thousands of instructions per second, they are potentially capable of measuring the times of various events to a resolution of a few microseconds. The 6522 versatile interface adaptor (VIA), which we have been looking at previously, has two 16-bit counter/timers and is therefore ideally suited for measuring time.

As in previous articles in this series, we will be looking in particular at the BBC microcomputer, but the principles are equally applicable to many other 6502 based systems. The BBC microcomputer has quite a number of different means of measuring time, and all are based in one way or another on one of the computer's two VIAs.

Time

The simplest timing facility is the BASIC pseudo-variable "TIME", which records the time in centiseconds. The user can set the time value by using a "TIME=" assignment and can read it with "variable = TIME" or "PRINT TIME". The BASIC interpreter is in fact making use of one of the timing facilities provided by the operating system. This is what the User Guide refers to as the "system clock", and is a variable which is up-dated under interrupt every 10 milliseconds (= 1 centisecond). Unless it has been altered by the programmer, the system clock and hence the value of TIME, gives the length of time since power-up or since the last hard reset, ie CTRL break.

The system clock cannot only be accessed from within BASIC using TIME, but also from machine code programs by using two of the operating system calls directly (OSWORD 1 and 2). There is also another clock on the BBC micro, known as the "interval timer", which is accessible through the operating system, though not directly through any BASIC function. This value of time is also measured in centiseconds, but is entirely independent of the system clock. That is to say, reading and writing the interval timer has no effect on the value of TIME, and vice versa.

Each of these timers is represented by a five byte integer and is therefore capable of recording a time period of up to 2^40 centiseconds which is about 340 years. In BASIC however, integer variables are only four bytes long, and so TIME, to be compatible, is also only four bytes long, the fifth byte of the system clock simply being ignored. This means that in BASIC, you can "only" measure times of up to a maximum of 16 months!

Program 1 gives an example of how the interval timer can be used in machine code programs. The technique is to save yourself a bit of memory, referred to as the parameter block (PBLOCK% in this example), for the operating system to use when transferring the data to and from the program. The routines used here are OSWORD 3 and 4 for reading and writing.

PROGAM 1

To illustrate the use of OSWORD routines to read and write the interval timer.

```
 10 CLS
 20 DIM CODE% 20, PBLOCK% 5
 30 OSWORD = &FFF1
 40 P% = CODE%
 50 [ OPT 2
 60
 70 .write
 80 LDX #PBLOCK% MOD 256
 90 LDY #PBLOCK% DIV 256
100 LDA #4
110 JSR OSWORD
120 RTS
130
140 .read
150 LDX #PBLOCK% MOD 256
160 LDY #PBLOCK% DIV 256
170 LDA #3
180 JSR OSWORD
190 RTS
200 ]
210
220 !PBLOCK% = 0
230 CALL write
240
250 T1% = TIME
260 PRINT '"TIME is now ";T1%
270
280 INPUT '"Delay until RETURN" W
290 CALL read
300
310 T2% = TIME
320 PRINT '"TIME is now ";T2%
330
340 PRINT '"Change in TIME = ";T2% - T1%
350 PRINT '"Interval timer = ";!PBLOCK%
```

The value of time obtained from the machine code part of the program is compared in the BASIC part of the program with the value of TIME, and there is never more than a centisecond difference. We will not go into any further details here since it does not relate directly to the use of the VIA timers, and is covered in the User Guide, pages 458-460.

Using the 6522 timers

If making a measurement in centiseconds does not give sufficient resolution, then the timers on the 6522 VIAs can be employed directly. There are two 16-bit timers on each of the VIAs, but both the timers on the internal VIA are being used by the operating system and should be left well alone. In fact it is Timer 1 on the internal VIA which is used to generate the centisecond interrupts used for up-dating the system clock and the interval timer. Timer 2 is used for controlling the timing of the internal speech generation system.

The original intention of the designers of the 6522 VIA was that the counter/timers would be used for generating time delays rather than measuring time. The idea was that a two byte number would be loaded into the timer which would then decrement to zero, set a flag and, if required, generate an interrupt. So if they are to be used instead to measure time then the course of events has to be as follows.

1) Wait for the start of the event to be timed.

2) Load the timer with a large number, probably &FFFF.

3) Wait for the event that signals the end of the time period.

4) Read the current value held in the timer.

This technique is illustrated in **Program 2** which measures the time period of a continuous train of pulses applied to PB7 on the User Port. One problem which occurs is that when you come to read the timer value, you are trying to read a 16-bit value, but the processor can only handle 8 bits of data at a time, so you can get an erroneous reading if the high byte changes after you have read the low byte.

It is similar to the problem that occurs when writing to the timers. That is solved within the VIA itself, as we saw last month, by each timer having a latch which is used to store the low byte of the data. Then when the high byte of the timer is written, the contents of the latch are transferred into the timer and the timing starts. Unfortunately there is no means of reversing this process.

PROGRAM 2

To measure the time period of pulses coming in on PB7 of the 6522 VIA.

This measures the time to a resolution of 4 microseconds in a period of up to 131 milliseconds, measuring from negative edge to negative edge.

```
 10 CODE%=&C00 :REM or wherever you want it
 20
 30 PB = &FE60
 40 T2L = &FE68
 50 T2H = &FE69
 60 ACR = &FE6B
 70
 80 P% = CODE%
 90 [OPT 2
100
110 SEI          \ Don't interrupt the timing!
120
130 LDA #&20   \ Set Timer 2 to count
140 STA ACR    \ PB6 pulses, i.e. stop
150
160 LDA #&FF
170 STA T2L    \ Set Timer 2
180 STA T2H    \ to &FFFF
190
200 LDA #0     \ Ready to start Timer 2
210 LDX #&20   \ Ready to stop Timer 2
220
230 BIT PB     \ If PB7 = 0, wait.
240 BPL P%-3   \ i.e. until PB7 = 1
250
260 BIT PB     \ If PB7 = 1, wait.
270 BMI P%-3   \ i.e. until PB7 goes to 0
280
290 STA ACR    \ Start Timer2
300
310 BIT PB     \ If PB7 = 0, wait
320 BPL P%-3   \ i.e. until PB7 = 1
330
340 BIT PB     \ If PB7 = 1, wait
350 BMI P%-3   \ i.e. until PB7 goes to 0
360
370 STX ACR    \ Stop Timer2
380
390 STA &406   \ Zero in 3rd byte of A%
400 LDA #&20   \ Check whether Timer 2
410 BIT &FE6D  \ timeout flag is set.
420
430 BEQ P%+2   \ If not, jump over INC.
440 INC &406   \ INC 3rd byte of A%
450
460 LDA T2L    \ Get low byte of count
470 EOR #&FF   \ Subtract from &FF
480 STA &404   \ Store in low byte of A%
490
500 LDA T2H    \ Get high byte of count
510 EOR #&FF   \ Subtract from &FF
520 STA &405   \ Store in next byte of A%
530
540 CLI        \ Now you can interrupt!
550 RTS
560 ]
570
580 MODE7
590 REPEAT
600   CALL CODE%
610   PRINT A%
620   UNTIL0
630 END
```

However there are two possible techniques which can be used in order to read the 16-bit value correctly. The first is to stop the timer altogether so that we can read it at leisure. This is illustrated in **Program 2.** The second method is to read it even though it is changing, but to read the low byte first and then make adjustment for the fact that at certain values of the low byte (2, 1 or 0), the high byte will have decremented before we get time to read it. This is illustrated in **Program 3** (see lines 830-940).

Stops and starts

In **Program 2,** the timing is done by using Timer 2. The reason for this is that Timer 1 can never be stopped since it is always fed with 1MHz clock pulses. Timer 2 on the other hand can either receive 1MHz clock pulses or pulses coming in on PB6. Thus if there are no pulses coming in on PB6 and we switch over to it, then the timer stops altogether.

The timing sequence in **Program 2** then is to switch to PB6 ie stop the timer (lines 130, 140), load it with &FFFF (160-180), wait for a negative-going edge on PB7 (230-270), switch the 1MHz pulses on (290), wait for the next negative-going edge (310-350), and switch 1MHz pulses of (370). Then, having switched the timer off, you can read the values at leisure.

Before you read the timer bytes, you have to check whether the Timer 2 time-out flag is set (lines 400-430). If it has timed out, it means that the counter has done one complete &10000 plus the value given by the two timer bytes. This is taken care of by putting zero in the third most significant byte of the integer variable A% (390) and then incrementing it if the time-out flag was set (440). This checking has to be done before the timer bytes are read, since reading the low byte of the timer causes the flag to be reset to zero.

The values of the two timer bytes are then read, but before each is put into the appropriate byte of A% it has to be subtracted from &FF since the timer was counting down from &FF. This is done by exclusive-OR'ing (EOR) with &FF which has the same effect as subtracting from &FF.

This whole process is done with interrupts disabled (110, 540) since if an interrupt occurred just as the negative edge was about to arrive, it would not start (or stop) the timer until after it returned from the interrupt service routine.

The resolution of the timer is set by the fact that it takes 4 microseconds to check one of the port lines for an event. Thus the time is measured to the nearest 4 microseconds. However if it is a repetative pulse, you could take a series of readings and average them. This means that, depending on where in the cycle the timing happens to start, the result will be the nearest value above the actual value, or the nearest below. By averaging, you get a weighting towards the value which occurs more often thus effectively splitting the 4 microsecond gap and improving the resolution.

Measuring single pulses

If you are trying to measure, say, the time taken for an object to cut through a light beam, then you only get a single pulse and so cannot use averaging unless you can get the event to re-occur. More importantly you have to change the "waiting" part of the program. For example, if the light beam system gives a positive-going pulse, ie normally zero, then you want to wait until

PROGRAM 3

To measure the time period of pulses coming in PB7 of the 6522 VIA.

This measures time with a resolution of 4 microseconds in a period in excess of 131 milliseconds, up to a few seconds, measuring from negative edge to negative edge.

```
  10 CODE%=&C00:REM or wherever you want it
  20 FROCassemble
  30 T%=10000
  40
  50 REPEAT
  60   base%=&10000
  70   A%=0 : REM start interval timer
       at zero
  80   CALL rough_time
  90   R%=A%*10000 MOD base%
 100   IF R%<T% OR R%>(base%-T%) base%=50000
 110   X%=(base% - 2) MOD 256
 120   Y%=(base% - 2) DIV 256
 130   CALL fine_time
 140   ti%=A%*10000 DIV base% * base%
 150   time% = ti% + base% - B%
 160   PRINT"time%= ";time%
 170   UNTIL0
 180
 190 DEF FROCassemble
 200 B% = 0
 210 PB = &FE60
 220 T1L = &FE64
 230 T1H = &FE65
 240 ACR = &FE6B
 250 OSWORD = &FFF1
 260 parameter_block%=&404 : REM A%
 270
 280 FOR N% = 0 TO 2 STEP 2
 290   P% = CODE%
 300   [OPT N%
 310
 320   .rough_time
 330
 340   BIT PB     \ If PB7 = 0, wait.
 350   BPL P%-3   \ i.e. until PB7 = 1
 360
 370   BIT PB     \ If PB7 = 1, wait.
 380   BMI P%-3   \ i.e. until PB7 goes to 0
 390
 400   .write
 410   LDX #parameter_block% MOD 256
 420   LDY #parameter_block% DIV 256
 430   LDA #4
 440   JSR OSWORD
 450
 460   BIT PB     \ If PB7 = 0, wait
 470   BPL P%-3   \ i.e. until PB7 = 1
 480
 490   BIT PB     \ If PB7 = 1, wait
 500   BMI P%-3   \ i.e. until PB7 goes to 0
 510
 520   .read
 530   LDX #parameter_block% MOD 256
 540   LDY #parameter_block% DIV 256
 550   LDA #3
 560   JSR OSWORD
 570
 580   RTS
 590
 600   .fine_time
 610
 620   SEI        \ Don't interrupt the timing!
 630
 640   STX T1L    \ Set timer 1 low latch.
 650
 660   LDA #&40   \ Set Timer 1 to free-run.
 670   STA ACR
 680
 690   BIT PB     \ If PB7 = 0, wait.
 700   BPL P%-3   \ i.e. until PB7 = 1.
 710
 720   BIT PB     \ If PB7 = 1, wait.
 730   BMI P%-3   \ i.e. until PB7 goes to 0.
 740
 750   STY T1H    \ Start Timer 1
 760
 770   BIT PB     \ If PB7 = 0, wait.
 780   BPL P%-3   \ i.e. until PB7 = 1
 790
 800   BIT PB     \ If PB7 = 1, wait.
 810   BMI P%-3   \ i.e. until PB7 goes to 0.
 820
 830   LDX T1L    \ Catch low byte first,
 840   LDY T1H    \ then high byte.
 850   CPX #3     \ Check if it's 2, 1 or 0.
 860
 870   BCS more_than_two
 880
 890   INY        \ T1H must have decremented
 900              \ AFTER T1L was read.
 910
 920   .more_than_two
 930   STX &408   \ REM = B%
 940   STY &409
 950
 960   CLI
 970   RTS
 980   ]
 990   NEXT
1000 ENDPROC
```

the input is logic 1, start the count, wait for it to go to zero again and then stop the count. To do this, simply omit lines 260, 270, 310 and 320. For a negative-going pulse, omit lines 230, 240, 340 and 350 instead.

The maximum time which this program can measure is &1FFFF microseconds (131 milliseconds or a frequency of about 7Hz). If the time which is to be measured is greater than that, the timer will go round a third time, and since the time-out flag is only set once, it will be impossible to tell whether the time was &1FFF, &2FFF or &3FFF etc.

"Timer 1 in free-running mode will automatically re-load the timer . . ."

Extending the time range

One method of extending the range of the time measurements which I looked at, was to check the time-out flag each time the port line is checked and if the time-out flag is set, increment a counter, for example the low byte of the integer variable B%, at &408. If you use Timer 1 in the free-running mode, it automatically re-loads the timer each time it times out. The idea seems a good one in theory – using a single byte counter in fact extends the range of the measurements to 16.7 seconds. However, having tried it, I found that if the time being measured happened to be approximately a whole number of "times round" the VIA timer, it gave an error of 65 milliseconds, and I could not find any simple way of detecting exactly when this error would occur in order to make the necessary correction.

The alternative method which I then developed is illustrated in **Program 3.** First of all you take a rough measurement using the operating system's interval timer. This allows you to estimate how many times the timer will go round, and you therefore do not need to keep track of how many times the timer does go round while making the actual reading. Also you can check whether, using the initial value of time to be loaded into the timers, it would come too near to being a whole number of "times round". If so, the number to be loaded into the timers (base%, lines 60-100) can be changed to a different value (50,000). The value of R% at line 90 gives an indication of approximately what the two bytes of the timer would read if 65536 (&10000) were used. If this value is either less than 10,000, or within 10,000 of the maximum count, the value of base% is changed to 50,000.

This appears to work very effectively, even when time values reach the order of seconds. Mind you, at that sort or level, the generator producing the pulses under test is likely to be at the limits of its stability. On the tests I did, using times in excess of a second, the value displayed by the program did tend to vary by more than the 4 microseconds that occurred at shorter times. This is not surprising though since we are only talking about variations of a few parts per million. This will, I am sure, be adequate for most purposes.

Having made claims about resolutions of a few parts per million with these timers, it is worth asking whether these are significant in the light of the overall accuracy (or inaccuracy!) and stability of the measurements. The absolute accuracy depends on the frequency of the 16MHz crystal clock. As the computer warms up this varies by the order of 1 part in 5,000. However once it has warmed up, after an hour or so, the variations are generally less than 1 part in 100,000, but the actual value of the frequency can be of the order of 1 part in 1,000 out. Thus if absolute accuracy is important, then you will need to find a frequency meter against which you can calibrate, and add a correction factor at the stage before you display the result of the measurement.

Next month. How to use the BBC micro as a note ranging frequency meter, 7Hz to 400kHz with an accuracy of better than 1 part in 50,000.

Diary of a QL user

When is a QL not a QL? Answer: when it's a PM, FB or EL. These are just three versions of the QDOS operating system that have infiltrated their way into the Sinclair machine.

It is now common knowledge that when the QL was launched on January 12 the machine was not fit for sale, and nor would it be for some months. Numerous attempts were made by the Company's engineers to correct the many QL teething troubles, and it appears that they have now largely succeeded. However, early QL buyers were made to pay the price for the premature launch and some contentious advertising claims. They received machines riddled with bugs of both the software and hardware variety, and in the words of Sir Clive Sinclair himself, 'we are getting them to find those bugs for us'*

***E&CM* reviewers have done their own fair share of bug-hunting, but now perhaps it is time to take a hard look at the birth pains of the QL. Below we publish the full and we hope final version of the QDOS story, and the diary of a QL user who found one bug too many, as well as Sinclair Research's own reply.**

**quoted in Personal Computer News 16 June 1984*

. . . 12 January

Press launch of QL computer.

. . . 13 January

10.00 am order by telephone using Barclaycard

. . . 3rd May

QL arrives by post. *FIRST IMPRESSIONS:*

> The keyboard is awkward to use, long key travel and noisy, not at all a "professional keyboard".

> The three plastic wedges which angle the keyboard are retained by ½mm of rubber foot, and fall out all the time. They have no rubber on their base.

> The base of the case is warped, so it rocks laterally on its six rubber feet as I type.

> The width of the computer picture required internal adjustments to both Philips and Thorn 12 inch monochrome TVs to allow use even in 40 character "TV mode". In 80 character "Monitor mode" the edges of the text screen are within the line flyback period, so are invisible.

> No manual on BASIC or the operating system, only for the 4 Psion programs, with conflicting loading instructions, one of which causes the machine to lock up. "Don't worry – it will arrive by post in the next few days" says the accompanying letter.

> No screen editor on this price machine! Line editor of Basic programs unbelievably poor. After typing in a long line with a simple mistake, the remark "bad line" is given and you have to retype it all. (Has the designer seen the Sinclair Spectrum?).

> Free gift of an RS232 lead! That's just as well as those British Telecom looking plugs from BICC have a different keyway polarisation which is not yet on sale. Cambridge cannot tell me where to buy one, so the "gift" is the only way to use the RS232. £14.95 would be some price to actually pay.

> Microdrive number 1 is only recognising that a cartridge is inserted 50% of the time. This is a major bore, all that typing of the long winded commands has to be done again and again.

> The 4 Psion programs: Quill and Abacus fail to LOAD even from their Master copies, grinding the cartridges on and on for several minutes before halting with the curt message "bad medium", presumably referring to the tape.

> With the 2 Psion programs which do LOAD, I am surprised to find them using "overlay" program sections, causing a considerable

The QL and the ver$ story

Typing PRINT ver$ into any QL is rather revealing. This little statement prints the version of the EPROM squatting at the back of the machine.

The first to hit the streets (all 300 of them) were version FB. This is the most dire and unpleasant version you can have (see above).

Not so much bugged as delightful if you find something that actually works; this was rushed out to hopefully calm the mounting storm of soon-to-be customer dissatisfaction. It didn't help.

Prevalent bugs in this machine were things like –1–1 returning a result of 0, microdrives taking a weekend in Spain for two before actually returning with their intended job done, and more crashes per second than Barry Sheen at his best.

Things had to improve. Version PM was to provide the next solution. This collection of EPROMs (now we know where all the 27128s went!) is much closer to a real answer. Although being recognised as not the perfect operating system or SuperBasic, it is a great deal closer than the previous firmware. The single most noticeable upgrade is the Basic line editing. Version PM offers some, while the earlier FB is annoyingly tedious to use.

FB, PM and AH (see below) are not the only versions of QDOS in existence. Some lucky (or not so lucky QL owners will have the EL and EK (or is it KB?) versions. We know little about these, but confidently predict that such QLs will become the penny blacks of the computer world.

On the Spectrum, entering lines of a Basic program was a joy once you had got used to the not exactly revolutionary keyboard. You could type a line that was full of syntax errors knowing full well that the line would not be accepted by the Spectrum when you finally pressed ENTER. You could also bring a Basic line down into the editing area of the Spectrum by pressing the EDIT key, which would then allow you to alter any aspect of the line by simply moving the cursor to the relevant place and inserting or deleting the requisite text.

Not so with the FB version of the QL operating system. All syntax errors or mis-typings (and no-one is a perfect typist) would simply result in an error message being displayed. You were not given the chance to make instant amendments. You had no choice but to retype the whole line. With some of the QL's device syntax this was never much fun.

Thankfully the second version of the operating system cured all this. Sinclair brought back its much loved on-entry syntax checking with a vengence. Not only is the error message printed, but the line is returned to you with full editing capability. The only problem with this part of delay. I understood that the QL approach was of RAM based data handling for speed. The microdrives are understandably slower than disc drives.

> The accompanying letter states:

VERY, VERY IMPORTANT. We VERY STRONGLY RECOMMEND . . .

and goes on to describe how to make backup copies of the Psion programs, preserving the Master copies. There are 8 blank cartridges supplied, plus the 4 Psion programs, the total of 12 cartridges would "retail" at an absurd £59.40.

Distinctly uneasy feeling about Sinclair's confidence in their microdrives is rising.

. . . 4th May

> Phone "QL Engineer" on the Cambridge number provided in the letter. Mr. Crabtree promises to send replacement programs, and asks for the defectives to be returned to him at Cambridge. He confirms that the editor is "not really acceptable" and the ROM upgrade in a few weeks time will be better.

. . . 5th May

> Use the QL to list its entire 42K ROM (yes 6K of empty ROM) to the screen to identify the Basic language command words.

The error reports are terse, and too few of them, I found 21 compared with the Spectrum's 48 informative messages.

I wonder why they chose DIR MDV1– when the much shorter CAT1 was used on the Spectrum to obtain the cartridge contents? Hewlett Packard uses CAT without feeling coy. If these machines are going to be networked a common syntax would be helpful.

> Why does the QL keep locking-up without warning both in simple Basic programs and in the Psion programs? The occurrences seem almost every hour now.

. . . 6th May

> Contact a past colleague in electronics who freelances for the computer press, and a current colleague, both QL "waiters" since January, and both still QL'less. My freelance friend brings a "provisional manual" given out at the press launch, what luck! No, almost everything has been changed since then it seems, only the simpler commands are unchanged.

> Multitasking, the real "quantum leap" at this price, is not implemented from Basic, as advertised. This is serious.

> Why is this 68000 processor engined computer so unbelievably slow? About twice as fast as the Spectrum (a slow computer).

> Try a prime number routine known as the Sieve of Eratosthenes, maybe its good at maths? Oh dear, it takes an hour, just like any other 8 bit Basic. I wonder why the 68000 based Hewlett Packard HP9816 running Basic does it in 4.5 minutes? The Spectrum takes just 22 seconds in Hisoft Pascal and the HP9816 takes 8 seconds. It seems to take the QL longer than 22 seconds to DIMension the array!

Oh look, if we set this BEEP statement after the DIMension, it BEEPS for ever, although due to

Are these the last bugs to crawl out of the QL? Almost – but not quite.

By Adam Denning

the system now is that the sheer superiority of the QL's SuperBasic makes it a little harder to detect syntax and parameter errors than it was on the Spectrum. This is because user-defined procedures take on the appearance of SuperBasic keywords, so a mis-spelt keyword could be (and sensibly, generally is) taken as a user-defined procedure. There obviously isn't much of a way around this.

Not only does version PM offer this simple advantage, but it also offers a whole lot more in the editing and creation of Basic programs. The keyword EDIT allows the programmer to specify (if he so requires) an increment, so that once a line is edited the next in sequence can also be altered by pressing the down arrow key rather than ENTER. It therefore becomes a sort of advanced AUTO, as the next line does not have to exist to be edited. Obviously if it doesn't exist when it is offered for editing, one may create it then and there. But there's more to it than that. Pressing the up arrow key brings the line **prior** to the current one down for editing.

Another oddity of both FB and PM is the way variables that haven't yet been declared are treated. If you try to print the value of such a variable to the screen all that is printed is an asterisk. Presumably this is part of the mechanism for coping with procedures in which the number of actual parameters passed is less than the number of formal parameters declared in the procedure's definition.

There is a 'bug' (this one is closer to a 'feature'!) in one of the applications packages supplied with the machine – Psion's Easel. When entering values into the program the use of a trigonometrical functions such as SIN or COS causes the machine to totally hang up.

Finally (or is it?) the latest version of QDOS, AH, is now being blown into EPROM and even, rumour has it, into ROM. The first versions of this OS are now being shipped out to customers, and visitors to the Earls Court Computer Fair would have seen massed ranks of QLs without the ghastly EPROM sitting in the back – this is the final final version. QL owners will be entitled to one upgrade of their machine to AH in ROM, but not to the any future upgrades (these will not be made for another year or so anyway – Sinclair engineers have had enough of fiddling about). FB owners will be the first to receive the AH upgrade.

This does not mean to say that all bugs have gone: there are at least three Non Implemented commands in AH QDOS; these are, WHEN, WHEN ERROR, and a facility intended to allow the initialising of all the elements of an array. QL owners can try the following brain teaser courtesy of Sid of Micronet:

```
10  DIM a(7)
20  a=3
```

the "power of two processors" we can meanwhile SAVE the program on Drive 2, pull out the power plug and try to persuade the cat back into the house!

> NETWORK to a Spectrum. We succeed in LISTing a QL Basic program to the Spectrum after some time guessing the syntax. If the Spectrum does not absorb the message the QL will not "BREAK" back to Basic!

The QL does not use the conventional ASCII 13 to signify Carriage Return at the end of each program line, instead of 10 which is ASCII for Line Feed. The consequence is that we can LIST the file to the Spectrum ONLY if we make the QL send a final PRINT CHR$ (13), to tell the Spectrum to close its file. We cannot display the program on the Spectrum screen because all those line feeds are ignored by the Spectrum, so all the program is in one very long line! If we print it via a Spectrum Centronics Port to an Epson printer all those line feeds do look funny, better switch off before the paper runs out! This Epson printer offers optional Line Feeds with Carriage Return, but not the reverse.

We write a Basic program on the Spectrum to scan through the entire file changing all those 10's to 13's. Done it at last!

> Success, the QL adverts are right on this feature: "A long program runs as fast as a short one". This means that looping structures are absolute memory addressed, not by line number. The Spectrum slows down a FOR-NEXT loop even if only REMark lines are in front. (Hewlett Packard implemented this on the venerable 8 bit HP85 Basic).

. . . 8th May

> The replacement programs arrive, all four! I now have 16 cartridges (£95.20 "worth") and my son's two Spectrum cartridges look lonely, but I post the "defective" Psion set to Cambridge as promised.

. . . 11-13th May

> Spend the weekend trying to get some familiarity with the Psion programs. They look very promising, except for some silly places such as when trying and failing to READ something on the microdrives; the message "press and key to retry, or ESCAPE to abort the program". Thanks QL, that's two hours work scrapped at a stroke!

. . . 14th May

> Post QL to Cambridge.

Sinclair Research replies

Your correspondent's letter about the QL raises a number of questions in two distinct areas, customer service and technical specification.

As regards customer service, we do genuinely regret any delays which he may have experienced and any inconvenience and frustration arising. We are constantly reviewing and seeking to improve our customer services and are confident that his will not be the general view.

In the technical area many of his comments appear to reflect either an individual experience or indeed an individual preference. As a whole, with the QL, we sought to balance what was desirable and feasible from our point of view with what was required by and would be acceptable to the market.

Clearly no one specification can possibly satisfy every possible customer or application.

We believe that the QL provides a very powerful solution which opens up the possibilities of professional computing to a large new audience. On some specific points:

1. **Considerable care has been taken with the ergonomic features of the keyboard and they have proved highly acceptable to most reviewers. The wedges will not fall out if properly screwed in.**
2. **There was some delay in forwarding comprehensive manuals. All owners will now have received one.**
3. **We felt that the gift of the RS232 lead was particularly appropriate in that many QL users would be likely to require a printer facility very quickly. Understandably, its precise value will vary from user to user.**
4. **The use of 'overlay' program sections has not surprised any of the professional users. Even with 128K RAM available, powerful programs need to utilise them.**
5. **SuperBASIC has been designed as an extremely powerful programming tool. Speed is not the sole criterion in such a development.**
6. **The multi-tasking facility referred to is available.**

Sinclair Research
Stanhope Road
Camberley
Surrey

. . . 17th May

> Telephone Mr. Crabtree to check receipt. Had he been able to verify the problems? Not very informative but replacement machine will be Dataposted tonight. He has personally checked over the replacement machine, and copied the master programs over to the working cartridges to ensure no problems. The "microdrive mod" has been done, (subsequently found out this means bending the cartridge retention spring to hold it firmer against the head). He tells me that there are a number of bugs in Basic which will all go when the firmware goes into Mask ROM, this is happening now. This sounds much better, with a genuine attempt to respond to problems quickly. I am pleased, and tell him so.

. . . 22nd May

> QL arrived by Datapost, sent on 21st May.

> QL serial number different, and microdrive 1 recognises cartridges practically every time now.

> Why does the Space bar stick down, and why does that pretty L-shaped ENTER key bind unless pressed exactly in the centre? My freelance friend tells me has lubricated his keyboard, after pulling the keytops off. He says the keytop/spigot bearing surfaces of plastic scuff and bind. This keyboard photographs well, why does it not work well?

> Telephone Camberley to obtain Video pin connections. Connect the excellent Microvitec 895 dot Hi-Res colour monitor (at £400+ top of the range). Pin connection details wrong! Telephone Mr. Crabtree at Cambridge. Oh yes the pins have been changed since January. (Why does Camberley not know?). Telephone Camberley and suggest they contact Cambridge to get themselves organised!

Lovely picture, with overscan of 80 character (512 pixel) mode losing the edge two characters. Contact Microvitec, who tell me that they will produce a special version for the QL with a scan flyback time of less than 10 microseconds, the industry standard is 11.5 microseconds, as also used by the BBC and ITV for TV. What about existing schools monitors, mostly Microvitec under the DES scheme? That is a bit of a problem says Microvitec!

. . . 23rd May

Telephone Mr. Crabtree at Cambridge. There is a definite hostility to Psion voiced. It is suggested that the programs may be erroneously writing on their own program cartridge, removing the "write protect" tabs should prevent this. The Spectrum "write protect" is in software and is defeated by an "off the rails" computer with disastrous consequences. Mr. Crabtree tells me that for this reason the QL uses an electrical switch to prevent the write head being connected, even by an "off the rails" QL. The consequence is that the QL does not know whether the cartridge is "protected" and will happily appear to SAVE a Basic program, except the VERIFY routine built in will go on and on looking for the file, eventually "Timing out". This sounds pretty rough, but will the Psion programs work?

This theory appears invalid when I discover that the "non loading" programs actually do load most times if you leave them on. On the first QL the error report "bad medium" had appeared within a couple of minutes.

Spend three hours just loading and reloading the Psion programs. The 4 Masters take between 60 and 90 seconds to LOAD. The copies load in between 4 and 6 (yes 6) minutes.

I try a Spectrum trick of programming a FOR NEXT loop to repeatedly FORMAT a cartridge to stabilise the tape in the cartridge. The QL comes up with "format failed" after 8 loops!

A newly made copy of a Psion program loads in 2.5 minutes. Better but why not 60-90 seconds?

Modify the copy program called "clone" to swap the drives around during copying, no different.

> My colleague writes to Camberley cancelling his QL (due May).

. . . 24th May

> Telephone Mr. Crabtree at Cambridge. Why can this QL not work as a simple tape recorder? Mr. Crabtree blames the software.

> If the Master copies load in 60-90 seconds and copies load so slowly then surely the tape signal is poor. Do they know what magnetic flux level they are putting on the tape? ANSWER: NO.

> Since they have developed a tape recorder (the microdrives) someone must know what is going on? ANSWER: He is writing the code to analyse what is on the tape sectors at this moment!

> Have you used an oscilloscope to look at the heat signals? ANSWER: Yes, but too difficult to come to any conclusions, due to the speed of the data streams.

> I say I am losing patience, but as a practising professional Electronics Engineer my curiosity, together with a feeling for a fellow engineer, allows me to be persuaded to return the QL for yet another try.

> The manuals are still "being printed".

11.00 am

> Purchase the latest computer magazines. Read of other QL's having similar problems.

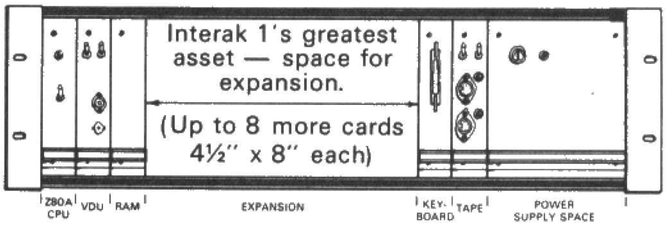
> I realise that I HAVE BEEN HAD. The QL sold to me is only a poor prototype, unreliable and with several important advertised features missing, yet I have paid £441.95.

I am not interested in waiting weeks more for the "improved" ROM firmware. I have been used as a development facility, at my expense, by an organisation which has shown a most cavalier attitude to their customers.

I have wasted an absurd amount of time. I also have a huge telephone bill.

> I telephone Mr. Crabtree for the procedure to obtain a refund. He suggests Camberley. I ask a lady at Camberley if I may return the QL by hand and collect a refund. She tells me that they have no one with the authority to raise a refund. I ask to speak to her superior. The superior is "unavailable". After some time I lose my cool and get very cross. Why she could not suggest I telephone Cambridge I do not know.

At Cambridge Mr. Crabtree suggests I ask for Sarah Johns of Customer Relations. After a frantic afternoon of telephone calls, it is arranged and at 5.00 pm I collect a full refund cheque from Camberley. The staff there had not seen a QL before, and I see some anxious faces, they know that all is not well.


Interak 1's greatest asset — space for expansion.
(Up to 8 more cards 4½" x 8" each)
Z80A CPU
VDU
RAM
EXPANSION
KEY-BOARD
TAPE
POWER SUPPLY SPACE

Centronics to RS232 converter

Micro manufacturers (especially the British) are notoriously fickle about the printer interfaces they use, choosing between one of two standards. T. E. Edwards has a way of overcoming the incompatibility problem.

There are a number of home computers available which have only a CENTRONIC interface for driving a printer. I suspect the reason for this is economy, as a serial RS232 interface is more difficult (ie more expensive) to implement. There are, however, quite a few 'hard copy' devices around which have serial interfaces. Some of these are available secondhand as firms up-grade their computer equipment. With the advent of the 'glass teletype' the real teletypes are becoming obsolete and it is possible to pick them up quite cheaply. It was with these hard-copy devices in mind that I designed this low-cost Centronics to RS232 converter.

The interface is not to the true RS232 specification as it only supports data flow in one direction, that is, outwards to a printer or other serial terminal. The circuit of the converter is shown in **Figure 1.** The most important item is the UART (Universal Asynchronous Receiver Transmitter) which is an LSI chip specifically designed for parallel to serial conversions. There are about as many different UART's as there are chip manufacturers but, except for one important difference, they all seem to have the same pin-out specification. UART's are not always compatible in that some require a −12V supply at about 10mA connecting to pin 2. The clever UART's make their own negative supply on-chip and require only one supply of +5V. **Table 1** shows the type numbers of suitable UART's for this design, with those that require −12V in addition to +5V are marked with an asterisk.

'The most important item is the UART'

To make this design 'low cost' a number of liberties have been taken. The BUSY line from the printer is conditioned to accept either RS232 or TTL levels by means of the diodes D1 and D2. If a TTL level is used the diodes do nothing, but if RS232 voltages are applied then diode D1 will conduct if the input line tries to go over +5.6V and diode D2 will conduct if the line tries to go below −0.6V. Thus the voltages are prevented from exceeding maximum TTL safe levels. There is no danger of damaging your printer drivers as specification for RS232 includes current limiting for an 'indefinite' time. There is a spare gate available on the 74LS132 and this can be used as an inverter in the BUSY line if your printer's busy signal goes 'high'. If this option is required, simply break the line at 'A' and connect the input from the diodes to pin 4 of the 74LS132 and pin 6 to pin 9. The most common sense direction for 'busy' condition is to go 'low' as shown. If your printer does not produce a 'busy' signal, or if you do not want to use it, just leave the BUSY input on the converter unconnected.

Another area of economy is the UART clock generator. The clock speed has to be 16 times the baud rate at which you want to transmit. I have used one element of the well used 74LS132 as a 'schmitt oscillator' but this has its frequency fixed by C1 and R so that unless switching is employed it is not easy to change baud rates. **Table 2** gives the required values of C1 and R to achieve any of the standard baud rates shown.

TABLE 1 List of suitable UART chips

AY-1013	*
TR1602	*
S1883P	*
COM8502	†
6402	†
6402-1	†

**Needs −12V on Pin 2*
†Pin 2 left unconnected

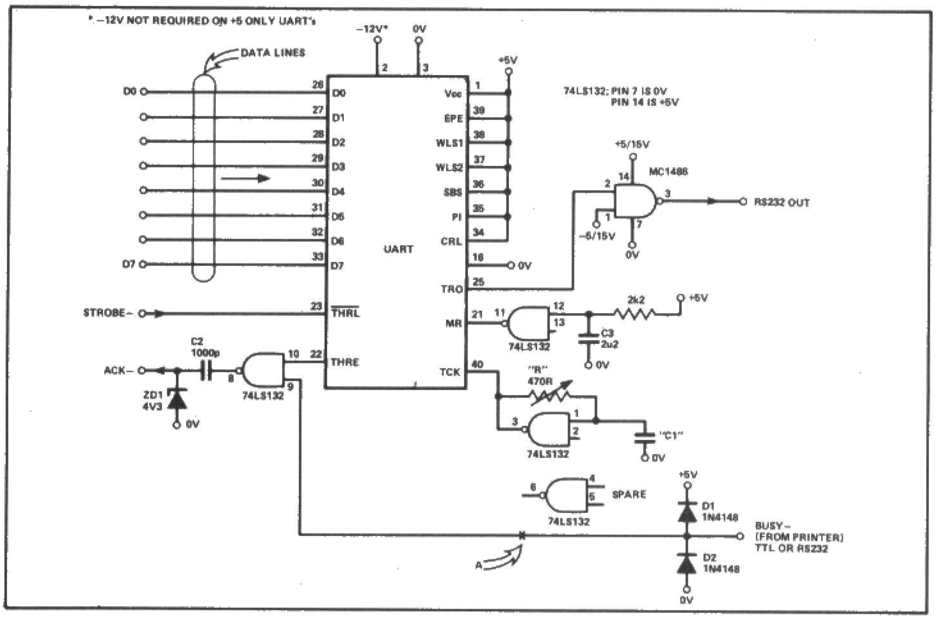


Figure 1. Circuit diagram of the converter.

It would be a shame to under-utilise our 74LS132, so the lost element or 'gate' has been pressed into service as an automatic reset generator. When power is first applied C3 is unable to charge up instantly, so the top plate which is connected to pin 12 of the gate stays low for a few milliseconds. This, when inverted, becomes a reset pulse for the UART. The 74LS132 is a very useful chip because of the built-in hysterysis it has on each input. This is what improves the reset signal and also allows us to use the chip as an oscillator.

TABLE 2 Baud rate component selection chart

Baud Rate	Freq.	Period	"R"	"C1"
110	1.76KHz	568.2μS	160Ω	2.2 μF
300	4.8 KHz	208.3μS	270Ω	0.47 μF
600	9.6 KHz	104.2μS	120Ω	0.47 μF
1200	19.2 KHz	52.1μS	120Ω	0.22 μF
2400	38.4 KHz	26.0μS	380Ω	0.047μF
4800	78.8 KHz	13.0μS	170Ω	0.047μF
9600	153.6 KHz	6.5μS	400Ω	0.010μF

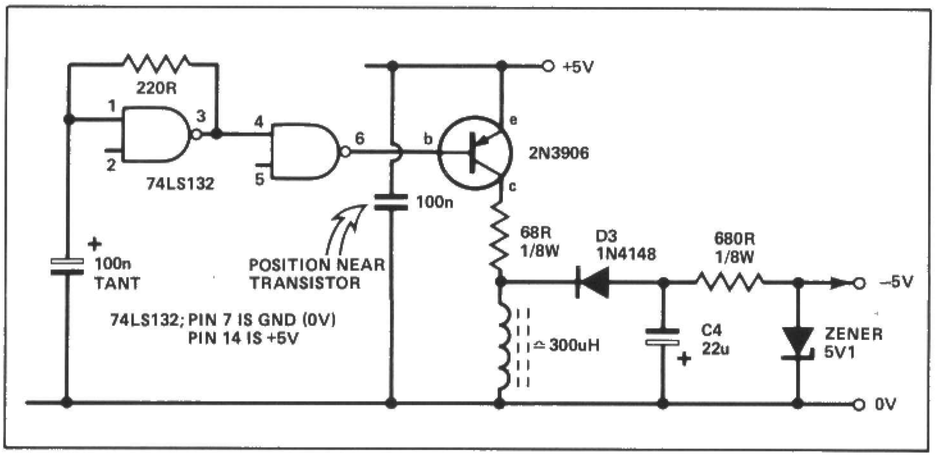


Figure 2. Diode pump circuit to generate –5V supply from a +5V rail.

Briefly, the Centronics interface convention is as follows: the computer puts the character on to the data lines as true data; that means a 'one' is represented as a high voltage. The computer then puts a low pulse out on the STROBE line for about 3 microseconds. It then waits until it sees a low pulse come back in on the ACK– line. This signals the computer that the character has been dealt with and that it can put another character out if it wants to.

The Centronics handshake is implemented by STROBE and ACK.

As shown with the zener diode Z1 and capacitor C2 in the circuit, the converter is designed to the correct Centronic specification but in practice I have found certain home computers do not work to this standard. In particular, the Dragon's Centronic interface has the ACK connected to its 'READY' line. This subtle difference means that the Dragon will not transmit any characters unless the ACK line is low to begin with. The removal of Z1 and the replacement of C2 with a shorting link cures this problem. With these components removed, I have found the converter works with computers whose Centronic interface is truly Centronics (BBC etc) so I advise you to omit the Z1 and C2 unless you really need a 'true' Centronics compatible interface.

The configuration of the UART has been set by the circuit diagram to give: 2 stop bits, no parity, and eight data bits. This will suit' just about everyone from teletype users upwards. There are not many devices which need 2 stop bits, but a printer which does require them will not work properly if it is given 1 bit, whereas a printer which is set for 1 stop bit will work correctly if it is sent 2. For the purists and those that need the slight extra speed however, if 1 stop bit is required then connect pin 36 of the UART to 0V instead of +5V. All the pins of the UART not shown on the circuit diagram are not required and can be left unconnected.

The serial output of the UART has to be converted from TTL voltage levels to RS232 levels. This is done in the MC1488 chip. The choice of either the chip or using discrete components was resolved by the current limiting advantages of this chip and its 'standardness'. The problem with all RS232 drivers is that they require a positive and a negative voltage supply. This should be between 6V and 15V, but I have found no configuration that did not work at +5V. If you have a plus and minus voltage available in your computer or in a separate power supply, then use that. Any voltage between 5 and 15 can be used but it is important that the plus and minus voltages are the same. If you use +12V then the other supply must be –12V. (Of course you will also need –12V if you use one of the UART chips marked in the table with an asterisk). However, if you only have +5V available on your Centronic output port as is the case with the BBC and Dragon, then it is possible to generate a –5V using another 74LS132 chip. **Figure 2** shows the circuit. Another oscillator is made using the 74LS132 which runs at approximately 3.5KHz. This feeds another element (to reduce loading) which drives a transistor. This transistor can be just about any PNP general purpose type. Do not be frightened by the coil! I used an RM6 type which is obtainable from Radio Spares, wound with as many turns of 0.5mm enamelled wire as I could get on it (about 50). This gave me a coil of approximately 300uH. The primaries of output transformers also work as a coil so don't be afraid to try anything you have lying around! The diode D3 is a 1N4148 but again, any old diode will do (not a zener of course!). This diode charges up the capacitor – watch the polarity! The resistor and zener make sure that the voltage is limited to about –5V. It is important that the zener is not left out whilst this circuit is driving the 1488 chip as the voltage would then be too high. Using this circuit to supply the negative voltage to pin 1 of the MC1488 means that the other supply pin (14) can now be connected to the +5V. There is about –20V or more at the top of C4 which the enterprising constructor could use to derive –12V for the UARTs requiring this. If this loads the negative supply circuit too much, a more efficient switching transistor can be tried. The price of the single supply UART is only a couple of pounds more than other types so this is probably not worth the effort. Decouple each chip with a 0.1uF capacitor connected from its power supply pins to 0V. This applies to both pin 1 and pin 14 in the case of the MC1488. Position the capacitors physically near the chips.

Setting up

The 'R' value is achieved by setting the variable potentiometer ('pot') R to the desired value shown in **Table 1** using a meter. Do this before connecting it in circuit. If you do not have any way of setting the value exactly, then wind the 'pot' to its maximum value before switching on and then carry out the ranging as follows: Connect your printer to the output of the converter. Switch on computer, converter and printer. The ranging adjustment should be carried out by anyone who does not have access to an oscilloscope or frequency meter. Decrease the resistance of 'R' by turning the 'pot' whilst the computer is continuously outputting characters. The printer will eventually start printing the correct characters as the baud rate zeros in. (Do not worry if the printer shows no signs of life when you begin). Make a note of where the 'pot' is and continue turning it in the same direction. When the printer stops printing 'good' characters, back up until it is just printing properly again. Now set the 'pot' to halfway between the noted points. A multi-turn 'pot' is the best type for 'R'. Do avoid reducing the resistance completely to zero as this may damage the 74LS132. If required, a manual reset push button can be connected across C3 so that it momentarily shorts out C3 when pressed.

'the interface is not the true RS232C spec., supporting data flow in only one direction'

MEMEX

After publication of the first part of our MEMEX project Cambridge Computer Consultants drew our attention to the fact that they had applied for patents in respect of their Aries board and that our design infringed these patent applications. We have therefore agreed not to publish part two of the project as planned but publish details of how Aries copyright software may be used with the hardware already described in *E&CM*.
Aries Computers, Science Park, Alton Road, Cambridge CB4 4BH. Telephone: (0223) 852614.

The system software is supplied in a sideways ROM as a service. Services are extensions of the operating system, usually with the job of looking after some extension to the hardware, for example, disc drives in the case of the DFS. They fit into the system in a way that allows them to be accessed from any language or application program, even if this is running in a second processor.

The B20 service controls the memory expansion hardware in two ways:

1 To replace the screen memory so that programs can use memory up to &8000 as if the screen did not exist.

2 To allow programs to switch between the extra memory and screen memory explicitly, for special purposes.

The first function works by intercepting all OS calls that make access to the screen, such as WRCH, the routine that prints a character on the screen. Most of the time when a program is running, the screen memory doesn't appear in the memory map, but when a screen access is requested, it is switched in temporarily in place of the program memory. This scheme is only possible because the code that writes to the screen lives in the OS ROM, not the program RAM. The switching is totally automatic and transparent, so the extra 20K can be used with BASIC, VIEW, FORTH, Lisp, BCPL etc. without any special action from the user.

This mode of operation covers almost all user programs, but some programs need to make direct access to the screen. In this case the program instructions that make the access must obviously be outside the switched area, ie below &3000 (or above &8000). The program must also select the screen explicitly, and a special Acorn-assigned OSBYTE/FX call has been provided to do this.

A further type of program, that includes games, is designed to use the unexpanded machine and makes direct access to the screen RAM. Such programs cannot take advantage of the extra RAM anyway, and 256 bytes of workspace that the service software uses must be freed. For these, a simple command will leave the machine as if the expansion were not fitted.

The service software has been written to be completely tube-compatible. Even where the program runs in a second processor and the usual memory conflict doesn't apply it gives more memory for other service utilities like *BACKUP and * COPY on the DFS.

Direct control

A special OSBYTE/FX call, number 111, has been assigned by Acorn for direct control of the state of the memory switching. It has three functions (which can be combined): set state, restore state and examine state. **Figure 1** shows the complete definition of the call.

A simple use of the FX call is to switch between two data areas: the extension RAM and the screen RAM left unused in MODE 7.

To use the maximum amount of RAM for data, HIMEM is set to &3000, and memory between PAGE (OSHWM) and &3000 is used for the BASIC program and dynamic variables. The BBC Micro RAM from &3000 to &7C00 is used as data area 0, and the expansion RAM from &3000 to &8000 is used as data area 1, giving a total of 39K for storing data. This is illustrated in **Figure 2.**

The BASIC indirection operators (?,! and $) are used to fetch and store data as normal, but in addition the program selects data area 0 and 1 using the FX call.

To switch to data area 0 (&3000 to &7C00) use:

*FX111,0

To switch to data area 1 (&3000 to &8000) use:

*FX111,1

As an example, we shall take the case of a scientific program that processes a lot of data. 36K of data has been generated by another program and stored as two 18K files on tape or disc. **Listing 1** loads these two files into the two data areas for processing.

The same FX calls are used in the processing section of the program to select the set of data the !,? and $ operators will access.

Extended addressing from BASIC

Listing 2 is a definition of a function that performs the selection automatically, so that when using the ? operator, the data areas appear as the continuous area of RAM from &3000 to &0BFF. The function takes an address in this range, selects the

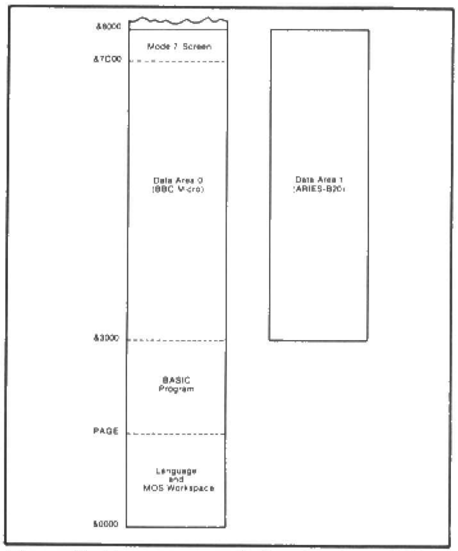


Figure 2. Memory map indicating how 39K of user RAM is made available.

appropriate data area, and returns a modified address. FNx is used with ? like this:

50 ?FNx(&C000)=?FNx(&C000)+1

Screen saving and loading

Saving or loading the screen with the system active is not as simple as it is on the unexpanded machine, since, when using the extra memory for program, the screen is not in the memory map. In BASIC, switching it in with *FX111,0 is not possible because the program and variables, and possibly the call itself, are then switched out. One answer is a machine-code routine which lives below &3000, and selects the screen while issuing a * command. **Listing 3** shows how this is done.

PROCoscrnass assembles the code. The code is 96 bytes long, starting at the address given as the argument. This can be anywhere below &3000, but below PAGE, in some unused bit of OS workspace, is best. &C00 is convenient if no soft characters are being used.

```
On entry:          A       &6F                                On exit:              undefine&
         X         determines function as follows:                        X         indicates state prior to the call
                                   Bit 6  0  write state of RAM select    1
                                          1  read state of RAM select                                  Bit 0  0  screen RAM
                                   Bit 7  0  no stack operation                                               1  program RAM
                                          1  read: pop state from stack           Y         preserved
                                             write: push state to stack           C,N,Z,    undefined
                   Bit 0   only used by write operations                          V,D       0
                                          0  select screen RAM                    I         preserved, and not altered during the call
                                          1  select program RAM                   B         preserved
```

Figure 1. Definition of the OSBYTE/FX III.

LISTING 1

```
 10 MODE7                     smallest screen mode is selected
 20 HIMEM=&3000               memory above &3000 is reserved
 30 *FX111,0                  select data area 0
 40 *LOAD FILEa 3000          load first file - goes to data area 0
 50 *FX111,1                  select data area 1
 60 *LOAD FILEb 3000          load second file - goes to data area 1
    .                         )
    .                         ) rest of program
    .                         )
```

LISTING 2

```
100 DEFFNx(ad%)
110    IF ad% < &8000 THEN *FX111,1
120    IF ad% >= &8000 THEN ad%=ad%-&5000:*FX111,0
130 =ad%
```

LISTING 3

```
9000 DEF PROCoscrnass(Q%)
9010 REM assemble ARIES screen OSCLI
9020
9030 LOCAL args,oscli,osbyte,argpt,send,sdloop
9040 LOCAL badargs,nargsok,dsok,dum
9050 args=&600:oscli=&FFF7:osbyte=&FFF4
9060
9070 P%=&A8:REM zero page variables in *COMMAND area
9080 [OPT 0
9090 .argptr OPT FNEQUW(0,0)
9100 ]
9110
9120 dum=FNoscrnsrc(Q%,0): REM assembly first pass
9130 dum=FNoscrnsrc(Q%,2): REM assembly second  pass
9140 ENDPROC
9150
9160
9170 DEF FNoscrnsrc(P%,opt%)
9180 [OPT opt%
9190 .oscrn  LDA args   \ number of arguments
9200         CMP# 1:BEQ nargsok
9210 .badargs BRK
9220         OPT FNEQUB(0,opt%)    \ error zero
9230         OPT FNEQUS("Bad argument",opt%)
9240         OPT FNEQUB(0,opt%)
9250 .nargsok
9260         LDA args+3    \ argument type
9270         CMP# 129:BNE badargs
9280 \ str$ variable
9290         LDA args+1:STA argptr   \ get BASIC string variable
9300         LDA args+2:STA argptr+1 \ pointer to string control block
9310         LDY# 3:LDA (argptr),Y   \ length of string
9320         TAX
9330         DEY:DEY:LDA (argptr),Y  \ address of string
9340         PHA
9350         DEY:LDA (argptr),Y
9360         STA argptr              \ pointer to string
9370         PLA:STA argptr+1
9380         TXA:TAY                 \ offset to byte after string end
9390         LDA #13                 \ string terminator
```

```
9400 .sdloop STA args,Y              \ copy string to buffer -
9410         CPY# 0:BEQ send
9420         DEY:LDA (argptr),Y
9430         CLC:BCC sdloop   \ relocatable unconditional jump
9440
9450 .send   LDA# &6F:LDX #&80        \ stack RAM state and set to screen
9460         JSR osbyte
9470         LDX# FNLO(args):LDY# FNHI(args)
9480         JSR oscli                \ send command
9490         LDA# &6F:LDX# &C0        \ unstack old RAM state
9500         JSR osbyte
9510         RTS
9520 ]
9530 =P%
9540
9550 DEF FNHI(W%) = (W% AND &FF00) DIV &100
9560 DEF FNLO(W%) = W% AND &FF
9570 DEF FNEQUB(B%,opt%):?P%=B%:P%=P%+1:=opt%
9580 DEF FNEQUW(W%,opt%):!P%=W%:P%=P%+2:=opt%
9590 DEF FNEQUS(S$,opt%):$P%=S$:P%=P%+LENS$:=opt%
9600
9610 DEF PROCoscrn(S$):CALL oscrn,S$:ENDPROC
```

LISTING 4

```
 10 REM Hardware function test for constructional project
 20
 30 MODE 3:HIMEM=&3000:G%=TRUE
 40 VRAM=&C000:PRAM=&D000  :REM RAM select addresses
 50 S%=ABS(RND)
 60 X%=RND(-S%)
 70 ?VRAM=0:?&7FFF=&AA
 80 ?PRAM=0:?&7FFF=&55
 90 ?VRAM=0:IF ?&7FFF<>&AA THEN PROCselect:END
100 ?&7FFF=&AA
110 ?PRAM=0:IF ?&7FFF<>&55 THEN PROCselect:END
120
130 FOR I%=&3000 TO &7FFF STEP 4:!I%=RND:NEXT
140 X%=RND(-S%)
150 FOR I%=&3000 TO &7FFF STEP 4
160   X%=RND:IF !I%<>X% THEN PROCram(I%,X%,!I%):G%=FALSE
170   NEXT
180
190 ?VRAM=0
200 IF G% THEN PRINT"OK"
210 END
220
230 DEF PROCselect
240 ?VRAM=0:PRINT"Select failure"
250 ENDPROC
260
270 DEF PROCram(addr%,in%,out%)
280 ?VRAM=0
290 PRINT "Failure at address ";~addr%;" in: ";
300 PRINT;FNhex(in%,8);" out: ";FNhex(out%,8)
310 ?PRAM=0
320 ENDPROC
330
340 DEF FNhex(N%,lth%):LOCAL H$
350 H$=STR$~N%
360 =STRING$(lth%-LENH$,"0")+H$
```

Commands

The software is activated on power-up and CTRL- BREAK, printing the famous "BBC Computer 52K" message (unless a second processor is fitted).

The *XOFF command de-activates it on the next normal BREAK, and *XON activates it in the same way. These only take effect when BREAK is pressed, because they change the amount of workspace used by the software, and services are offered this opportunity only when the OS initialises. The message:

ARIES-B20: Press BREAK

is printed to remind you. If no *X-command has been given since the last BREAK, then pressing BREAK leaves the system in the same state as before.

A special command, *XWORD, has been included at Acornsoft's request for use with VIEW A2.1. This version has a bug which stops it responding correctly to the amount of memory available, so *XWORD fudges the system to avoid this. With the system active, typing *XWORD enters VIEW directly. VIEW A1.4 does not have this bug and uses the extra memory when entered normally.

To save or load the screen, PROCoscrn is called with the command string as the argument, for example:

100 PROCoscrn ("LOAD picture 3000")

This loads a MODE 0,1 or 2 screenful.

Most of the code is to check and get the string argument from the CALL control block. The interesting bit is the routine 'send' at line 9450. The sequence of actions is as follows:

1 Select screen RAM, stacking old state. OSBYTE A=&6F, X=&80
2 Send the command string. OSCLI X,Y point to the string
3 Restore the old RAM state. OSBYTE A=&6F, X=&C0

The stacking function of the OSBYTE call is used to preserve the current RAM state without having to read it.

Test program

Listing 4 is provided to fully test the completed project. It tests that IC5 is selecting the RAM area properly, and that all locations in the extra RAM are functioning. The RAM is tested by writing a set of 4-byte random numbers to the whole 20K, and then reading them back and comparing them with the same set. The complete test takes about 20 seconds, after which "OK" is printed if no faults are found. If a mismatch of written and read data is found, the relevant values are printed out in hex like this:

Failure at address 3000 in: 1536A54F out: 77F54501

The most likely RAM faults are shorted or open address or data lines. Data bus faults are easiest to track down. A broken data line will cause the same bit of every byte to return 1 (or possibly 0), and two shorted lines will cause two adjacent bits to be equal whatever was written in. In the case of faulty addressing, the value read out of certain locations will be totally different from those put in, and the distribution of the addresses where this happens will indicate which line(s) are at fault. If a RAM chip is not working at all, possibly due to one of the outputs of IC9 not getting through, then the program will give a list of failures over a complete 2K block, for example &3000 to &37FC if IC13 is not being selected.

The last word

Mike James dots the Is and crosses the Ts in the final instalment of his micrographics series.

This month's article is the last part of the Micrographics series. The end is nigh – not so much because there is nothing else to say about graphics on micros – there certainly is – but from here on in it becomes much more specialised. There are ways of doing particular jobs which are simpler and faster than the general ideas introduced in earlier parts of Micrographics would lead you to believe. For example, no one would try to draw a three-dimensional bar chart using the general purpose 3D viewer program developed over the past few months – well you might but only if you really wanted to view it from a number of different positions. Unlike the general 3D viewer, graphs and charts are usually produced by special purpose programs which work a lot faster.

In addition to the specialisation introduced by particular applications there is the collection of tricks and clever methods that can be used with a particular computer to produce graphics that would otherwise be impossible. For example, it is possible to produce very fast moving graphics on the BBC Micro by changing the way colours are assigned to various areas of the screen – this isn't the usual blank-reprint animation cycle introduced in earlier parts of Micrographics but it does allow the BBC Micro programmer to produce effects that would otherwise be impossible. There is no doubt that until low cost graphics hardware becomes much more powerful than it currently is, these special tricks will continue to be extremely important in Micrographics. The trouble is that there is not much that can be done by way of explaining the ideas that lie behind them all and if you don't happen to use the machine that they apply to, then they are of no practical value to you. For this reason the last part of Micrographics deals with another general machine-independent topic – easy ways of producing 3D effects.

Easy 3D

Although the techniques for producing 3D displays based upon storing three-dimensional co-ordinates in a point file and then using a projection to convert these to points in two dimensions is perfectly general, it is too slow for many applications. In particular it is much too slow to use if you are trying to produce 3D animated displays. Even if you can afford the time to wait while the 3D viewer works out where everything should be, it is obviously inefficient to use it to draw the same three-dimensional background for a game for example. In both of these cases it isn't necessary to store the co-ordinates of all the points and lines that make up the image because they will only be used once. The obvious solution is to go back to the simplest methods of creating Micrographics – that is either to draw directly on the screen (using a light pen or joystick) or write programs consisting of graphics commands that produce the display that you want. For example, if you want to produce a wire frame cube on the screen you could use the 3D viewer program and enter the co-ordinates of the eight corners or you could use the point and line editor (*E&CM*February 1984) to draw a cube on the screen, or finally, you could write a short BASIC program that plotted the twelve lines that make up the edges of the cube. If you only want to produce a single view of the cube there is no point in using the 3D viewer – this is overkill. If you don't want to change the cube then you could use a slightly cut down version of the point and line editor but even this is overkill to a certain extent. In this case the method that most games programmers would use almost without thinking is the easiest – write a subroutine that will draw the cube.

'the last part of the series shows how to produce easy 3D effects'

Producing subroutines that draw theedimensional objects is easy as long as you are something of an artist. If you're not then, like me, you will have to work quite hard to produce the effects that you want. However, there are a few principles that artists treat as obvious that the rest of us can benefit from having made explicit! For example, it helps to realise that when you are drawing you are trying to produce a projection that corresponds to a three-dimensional scene. Once you realise this simple fact you should also notice that you can choose the projection that you are using. For example, you can draw a wire frame cube using either a parallel projection or you can shorten the more distant edges to produce a perspective view – it's up to you to decide which is more appropriate for your application. Of course sometimes anything other than a perspective projection results in a display that looks silly. The reason for this is that while parallel lines meet at a point in a perspective transformation, they remain parallel in a parallel projection! This can make quite a difference in some cases (but more of this idea later).

A 3D sketch pad

If you are trying to learn to draw in three dimensions then one of the most educational (and fun) programs that you can use is a 3D sketch pad. The idea of a 2D sketch pad is simple enough and familiar to most programmers – a graphics cursor is moved around the screen by means of the arrow keys drawing a solid line to mark its trail. To extend the idea to three dimensions simply means adding an extra pair of keys to control the movement of the cursor in and out of the screen. In other words the position of the cursor is now given by three co-ordinates X,Y,Z and the value of the third co-ordinate is changed interactively by pressing say the "I" key to move the graphics cursor Into the screen and "O" for Out of the screen. The problem is of course where to draw the graphics cursor on the two-dimensional screen. This problem has already been solved earlier in the series by the introduction of projections. For example, if you want to create a 'parallel projection' sketch pad, ie one that uses a parallel projection, then the position of the

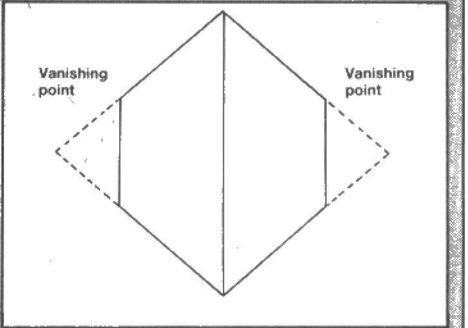


Figure 1. The vanishing point method of drawing a cube.

cursor X,Y,Z has to be converted to screen co-ordinates A,B using –

$$A = Y + U{*}Z$$

and

$$B = Y + V{*}Z$$

where U,V define the direction of the parallel projection (see *E&CM* May 1984). The best way to think about the way U and V control the projection is in terms of the way the x,y,z axis would be represented on the screen. As you would expect, the x axis is always horizontal and the y axis is always vertical but the angle that the z axis makes with the horizontal is given by the ratio of V and U. (In fact U/V is the TAN of the angle). The following program is a parallel projection sketch pad using U and V equal to 1 to make the z axis 45 degrees to the horizontal –

3D Sketchpad

```
10 MODE 0
20 U=1
30 V=1
40 X=500
50 Y=500
60 Z=0
70 *FX 4,1
80 A$=INKEY$(0)
90 IF A$=CHR$(136) THEN X=X-4
100 IF A$=CHR$(137) THEN X=X+4
110 IF A$=CHR$(138) THEN Y=Y-4
120 IF A$=CHR$(139) THEN Y=Y+4
130 IF A$="I" THEN Z=Z+4
140 IF A$="O" THEN Z=Z-4
150 A=X+U*Z
160 B=Y+V*Z
170 PLOT 69,A,B
180 GOTO 80
```

This program, like all of the short examples in this month's Micrographics has been written for the BBC Micro but, as none of the special features of BBC BASIC have been used, it should be easy to convert to other machines. Lines 90 to 120 form a fairly standard 2D sketch pad updating X and Y according to which of the four arrow keys are pressed. Lines 130 and 140 perform the update on the I co-ordinate, pressing the "I" key moves the current position further Into the screen and pressing the "O" key moves it further Out. The parallel projection is performed by lines 170 and 180 and finally line 190 plots the point at A,B. Using this program you should, after a little experimentation, find it possible to draw cubes and blocks without any trouble. Other shapes are possible but much more difficult. The program itself is capable of being expanded into something much more sophisticated but this is just a matter of putting it together with some of the ideas introduced earlier in the series.

The successful use of a parallel projection in a sketch pad program suggests that perhaps the perspective projection could be used to good effect. The perspective transformation is almost as easy to apply –

$$A = \frac{(XC{*}Z - X{*}ZC)}{(Z - ZC)}$$

$$B = \frac{(YC{*}Z - Y{*}ZC)}{(Z - ZC)}$$

where XC,YC,ZC is the position of the centre of projection. To try a perspective sketch pad make the following changes to the parallel sketch pad program –

```
 65  XC=500:YC=500:ZC=1000
150  A=(XC*Z-X*ZC)/(Z-ZC)
160  B=(YC*Z-Y*ZC)/(Z-ZC)
```

Line 65 sets the centre of projection roughly to the middle of the screen and on the viewer's side (positive values of Z move into the screen negative values out of it). Lines 150 and 160 perform the perspective transformation and replace the old lines 150 and 160. It takes a little while to get used to drawing with a perspective sketch pad so don't give up if your first attempts are confusing. It helps to remember that in a perspective projection lines appear shorter the further away they are – this also accounts for the way that the graphics cursor moves more slowly the further 'into' the screen it is.

Vanishing points

When you use the perspective sketch pad you can hardly miss the fact that one point on the screen is special because all of the lines that you draw by changing only the z co-ordinates converge on it. This is an example of a 'vanishing point', that is a point in a perspective projection where lines that are parallel in the three-dimensional space appear to meet on the two-dimensional screen. If you are trying to draw a perspective representation of a scene then it helps to know a little about the properties of vanishing points. In particular –

1) All parallel lines in a given direction in the three-dimensional space meet at the same point in the two-dimensional space – this point is known as a vanishing point.

2) Sets of parallel lines drawn in different directions on a plane all have different vanishing points but these all lie on a single straight line.

The first property is familiar to everyone who has looked at or tried to draw a ploughed field, for example, the furrows all head off into the distance toward a single point. The second property is less obvious but if you think of a scene consisting of a large field ploughed in different directions then you should be able to see that although the sets of furrows all meet at different points all these points lie on the horizon – that is on a straight line.

Using vanishing points to draw three-dimensional scenes is very easy once you have established the position of a vanishing point for each group of parallel lines used within the drawing. For example, if you are trying to draw a perspective view of a cube looking head on to one edge then there are two vanishing points that have to be fixed corresponding to the horizontal edges of each of the visible sides (see **Figure 1**). Varying the positions of these vanishing points produces different perspective views of the cube, however notice that there are some positions that do not produce a perspective projections of shapes other than a cube!

There are other properties of the perspective projection that are useful in drawing scenes. For example, the further away an object is the smaller it becomes and the more of them you can see. For example, in a field of flowers the closer flowers are larger and you can see fewer of them. Although this is obvious you might not have realised that the same idea can be used to give the impression of depth to a flat surface by varying the density of a random texture. For example, try –

```
10  MODE 0
20  FOR Y=600 TO 0 STEP -4
30  FOR X=0 TO 1280 STEP 4
40  IF RND(1)<Y/1200 THEN PLOT
    /69,X,Y
50  NEXT X
60  NEXT Y
```

You can also produce a depth effect by varying the size and shape of small objects drawn on a surface. For example, you could draw elipses that grow both smaller and thinner as they move further away from the viewer. Finally there are two distance effects that are not strictly due to perspective. The further away an object is the bluer and the fuzzier it appears. Most micros don't have enough colour resolution to give a smoothly graduated blue colouring to a scene but it is possible to make use of a rougher drawing of an object the further it is away.

3D billiards

If you are as hopeless at drawing as I am then you will prefer the slightly more complicated but always correct approach used in the next and final Micrographics example. Instead of either using a full 3D viewer program or completely abandoning projections in favour of methods based on vanishing points there is a compromise. There is nothing stopping you from using the equations for the perspective transformation to draw three-dimensional objects without using all the other methods used in the 3D viewer, that is point and line files, transformations etc. This is of course exactly the method used in the perspective sketch pad.

The key to using this method economically is to write a subroutine that will draw a line between a pair of points in three dimensions by way of the equations that define the perspective projection. Using this subroutine it is possible to draw representations of simple three-dimensional shapes by calling it repeatedly, in the same way that you would use a command that plotted a line between a pair of two-dimensional points to draw two-dimensional objects. This is best illustrated by way of a simple example.

The program given below first draws a perspective view of the top of a 'table' defined in three dimensions by the four points (50,100,0), (1000,100,0), (50,100,600) and (1000,100,600). As in the previous programs the z axis is taken to be at right angles to the screen and positive values of Z correspond to the 'inside' of the screen. After this the program then 'bounces' a small point around inside the boundaries of the table top in such a way as to give a correct impression to the viewer of a real ball moving in three dimensions on the surface. This sounds very complicated but it is just a straightforward application of the animation techniques based on sprites developed earlier in the series combined with the idea of perspective projection. The only real difference is that now the moving point, or sprite, has three co-ordinates and three velocities associated with it. In this particular case the y velocity is zero to keep the sprite on the surface of the table (that is it only moves in the x and z directions). Detecting a bounce on the table boundary is done in three dimensions and the appropriate velocity is reversed. After all the updating of position and velocity is completed, the perspective projection is applied to the sprite's position X,Y,Z to give its current position on the screen A,B. The final program is –

3D Billiards

```
  10 MODE 4
  20 XC=500
  30 YC=1000
  40 ZC=-1000
  50 GOSUB 1000
  60 GOSUB 2000
  70 X=500:Y=100:Z=300
  75 V=8:U=8
  80 GOSUB 3000
  90 GOTO 80

1000 X1=50:Y1=100:Z1=0
1010 X2=1000:Y2=100:Z2=0
1020 GOSUB 2000
1030 X2=50:Y2=100:Z2=600
1040 GOSUB 2000
1050 X1=1000:Y1=100:Z1=600
1060 GOSUB 2000
1070 X2=1000:Y2=100:Z2=0
1080 GOSUB 2000
1090 RETURN

2000 A1=(XC*Z1-X1*ZC)/(Z1-ZC)
2010 B1=(YC*Z1-Y1*ZC)/(Z1-ZC)
2020 A2=(XC*Z2-X2*ZC)/(Z2-ZC)
2030 B2=(YC*Z2-Y2*ZC)/(Z2-ZC)
2040 MOVE A1,B1
2050 DRAW A2,B2
2060 RETURN

3000 GCOL 0,0
3010 A=(XC*Z-X*ZC)/(Z-ZC)
3020 B=(YC*Z-Y*ZC)/(Z-ZC)
3030 PLOT 69,A,B:PLOT 69,A+1,B
3040 X=X+V

3050 Z=Z+U
3060 IF X<=60  OR X>=980 THEN V=-V
3070 IF Z<=16 OR Z>=580 THEN U=-U
3090 A=(XC*Z-X*ZC)/(Z-ZC)
3100 B=(YC*Z-Y*ZC)/(Z-ZC)
3110 GCOL 0,1
3120 PLOT 69,A,B:PLOT 69,A+1,B
3130 RETURN
```

Lines 20 to 40 set the location of the centre of projection, changing these values alters the view of the table top. Subroutine 1000 plots the table top by setting X1,Y1,Z1 and X2,Y2,Z2 to the co-ordinates of the pair of points that each line has to be drawn between. Subroutine 2000 is responsible for actually drawing the line after applying the perspective projection (lines 2000 and 2030). After drawing the table top lines 80 and 90 repeatedly call subroutine 3000 which animates the ball. The ball's initial position is set by line 70 to be roughly near the middle of the table and its initial velocities are set by line 75. Lines 3000 to 3030 blank the sprite at its old position. Lines 3040 and 3050 update its position by adding the appropriate velocities to its co-ordinates and then lines 3060 and 3070 check to see if it has reached a table edge. If so then the appropriate velocity is reversed to create a bounce effect. Finally lines 3090 to 3120 plot the ball at its new position. Notice the use of a perspective projection at lines 3010, 3020 and 3090, 3100.

Although in this simple example the motion of the ball is restricted to a flat surface it is not difficult to see how it could be extended to other more complicated forms of three-dimensional motion. You should also be able to see how subroutine 2000 could be used to draw perspective representations of more complicated objects. Finally this example is good fun to watch so it is worth trying.

Future projects

Micrographics is one of the most enjoyable of all areas of computing so although this series has drawn to a close I hope it has inspired you to carry on experimenting and developing techniques. Graphics are rewarding to the programmer because they really do provide a range of difficulty that can stretch your knowledge and ingenuity and because they are immediately appreciated by everyone. There are a great many worthwhile graphics projects that would fill gaps in the market or improve existing products. The whole area of interactive graphics is perhaps the most in need of some new ideas both to improve the user interface and overall efficiency. A three-dimensional graphics package is always a valuable project especially if you can find ways of including the more sophisticated techniques of hidden line elimination and shading of surfaces. Perhaps the most exciting new development is the availability of low cost video input devices. What exactly can be achieved using a TV camera and a computer is something that is a matter of imagination and technique – now is the time to start experimenting.

Improve your VIEW

Readers should refer to the Important Notice on the facing page.

Software Details

Where additional code is required to be added to VIEW, a JMP or JSR instruction which points at a jump table located at &B000 is inserted into the VIEW code. The jump table then directs the microprocessor to the correct piece of additional code. this technique was adopted to ensure that the JMP or JSR instruction inserted into VIEW did not have to be changed when alterations were made to the additional code.

In its unmodified state VIEW uses character values of &80 and greater to mark its edit command lines and rulers. The majority of the code alterations to VIEW are necessary to replace simple BMI or BPL instructions which branch if the top bit of the accumulator is set on clear respectively, with more selective versions. These are then able to distinguish between the new highlight characters, which have text values of &90 and &9E and the ruler and CL markers.

When insert mode is used near the end of a line and enough characters are inserted to exceed the line length, automatic reformatting of the line occurs. Unfortunately all the soft spaces in the line, ie those which are inserted by the word processor to create right justification, are inserted by the word processor to create right justification, are converted into hard spaces, giving multiple spaces between words which cannot be removed by reformatting. Line 1210 of **Listing 2** prevents this conversion taking place and eliminates the need to check the document for this problem.

Two further characteristics have been identified with regard to formatting. The first is that lines beginning with a tab character often have enough soft spaces to accept the leading word of the following line, but, upon reformatting, refuse to do so. The second is that highlight codes, although ignored in determining the number of characters which will fit on a line, are included in determining how many words will fit on to a line. These two characteristics are altered by the code at <formht>, <reform> and <tbform> in **Listing 1** (see *E&CM* July 1984), so that both tab characters and highlight codes are ignored when determining the number of words which will fit onto a line. A consequence of this is that the length of the line seen during text entry increases and sometimes may disappear off the edge of the screen when many highlight codes are used on a line. As is normal in VIEW, the end of the line may be seen by moving the cursor to that point. The disadvantage is that the whole of lines may not be visible without sideways scrolling of the screen. However, against this must be set the advantage of having better formatting with fewer spaces in the final document. The decision is a personal one

THIS LISTING CONTAINS MATERIAL WHICH IS THE SUBJECT OF A LEGAL DISPUTE

and if the new formatting features are not required then lines 1200, 1220 and 1230 of **Listing 2** should be deleted.

Built in Printer Drivers

The EPSON printer is reset to its power-up state before and after printing with USA and English character sets being selected when appropriate, to enable the pound and hash (#) signs to be printed. All the highlight commands act as toggles, that is the first occurence of the highlight enables the effect and the next occurence disables it. The subroutine which achieves this is <epseff> at line 3110. The subroutine expects the X register to contain the highlight code minus 128, this number falling in the range 0 to 13. The state of the highlights is kept in fourteen flags in page &400. If the X^{th} flag is zero then the effect is assumed to be off and so it is switched on. If the flag is non-zero then the effect is switched off. Switching is achieved similarly in both cases, by sending an escape character, CHR$(27), followed by the X^{th} character in <epon> or <epoff> followed by the X^{th} character in <epon2> or <epoff2>, if this is less than 128. Thus two or three character, control code sequences may be transmitted.

The JUKI driver operates in a generally similar manner but with the following differences. The driver will re-send the EMPHASISED or DOUBLESTRIKE commands, if these are selected, after a carriage return character has been sent, since the JUKI 6100 automatically disables these effects on receiving a carriage return. True super and sub scripts are created by moving the carriage half a line up or down as required. This is achieved by the code at <ef3>, line 2420 which changes the line spacing before moving the carriage and returns it to its original value afterwards. In some cases where an effect can be achieved with just one character code, as is the case with the backspace command, ASCII code 8, a harmless escape command is issued such as that to set the horizontal tab position before the backspace code is issued. If a highlight code greater than 140 is received by the printer drivers then an error message is displayed and printing stops. Due to the inherent construction of daisywheel printers, the JUKI 6100 cannot perform all the functions of which the EPSON dot matrix printers are capable. Thus some highlights such as ENLARGED, CONDENSED, ELITE and PICA only change the character spacing on the JUKI and not the size of the printed characters.

Other Printer Drivers

Any existing printer driver will work as normal, however if you have written your own driver then the following information will be of use to you in customising the driver to support the new features. The repeat flag, located at address &100, has the value &FF for the first and last calls to the printer on and printer off subroutines. After receipt of a call with this value, the value should be changed to zero so that subsequent calls may be identified as being other than the first or last calls. Highlight keys 1 and 2 generate the codes 128 and 129 as normal. SHIFT/CTRL and f0 and f9 generate the codes 130 to 139 and the backspace code from SHIFT/CTRL and cursor left appears as code 140. Superscript and subscript appear as codes 130 and 131 respectively, it is advisable not to alter this arrangement since these functions are reflected in the inverse characters printed in the text.

With care it is possible to customise the built in driver code so that printer drivers for printers other than EPSON and JUKI may be built in. The names of the two printer drivers are in a table at <pitable>, line 1560. Changing either of these names will alter the names to which the two built in drivers will respond. thus the JUKI driver can be converted to a SEIKOSHA driver by changing JUKI to SEIKOSHA at line 1580, altering the values in the JUKI on and off tables at lines 4040 to 4210 to the correct values to achieve the effects listed in **Table 1** and finally deleting lines 3800 to 3860 which only apply to the JUKI printer. The code between lines 3690 and 3720 will also need to be changed so that a pound sign is printed on the SEIKOSHA. It is most advisable not to attempt customisation until you have successfully

IMPORTANT NOTICE

Following our July article, Acorn have advised us that they consider it to be an infringement of the UK copyright in the VIEW program and it to be illegal to dump it from ROM or to transfer it to sideways RAM. We have therefore deleted the assembly language program that would allow this to be accomplished.

entered **Listing 1** and run the test option.

The first thirteen entries in the on and off tables should contain sets of values which enable and disable the functions given in **Table 1.** If your printer does not support a particular function, such as proportional printing, then insert harmless codes at the appropriate locations for this function. The fourteenth set of values should contain harmless codes if the printer moves the carriage up or down automatically in super and sub script modes. If this is not the case, as is true for the JUKI 6100, then these values should set the line height to an appropriate value so that a line-feed/reverse line-feed generated by codes 130 and 131 will place the carriage in a suitable position in which to print the subsequent characters.

Do not try to place more than 31 characters in the <prname> table at line 4480 since no more than this number of characters may be placed in the keyboard input buffer before values start to be lost. The names in the CL command table, <table>, may be changed if desired and this is located at line 2060. If a CL command is found which does not match one of those in the CL command table, then the unrecognised word is printed together with an error message and the PRINT, SHEETS or SCREEN command is terminated. The bit representations of the inverse characters are located at <invchars>, line 8210. These can be changed to more meaningful symbols, if required, to reflect changes in the CL command table.

Conclusion

The modifications provided in this article enhance the already powerful facilities provided by the VIEW word processor.

[redacted]

View printer driver generator

Acornsoft
Disc/cassette, £9.95

When the View word processor ROM was originally released by Acornsoft it had little facility for printing text, so Acornsoft quickly followed it up with a set of printer drivers for what they considered to be the most common printers.

This package has always been seen by users as inadequate.

It only covered a very few printers and in certain cases – notably the Epson FX80 – the most useful of the printer's features were not made available.

Hence the new package, View printer driver generator. This is available on both 40/80 track disc or cassette, and allows the user to generate a printer driver to his own specifications. Far more suitable and far more fitting of a product of View's versatility. As a sideline it also cures some of View 1.4's (the original version) more series 'features' such as disc-only printing capability.

The driver consists of what is effectively a simple machine code generator that takes parameters from the user and builds them into an assembly source file that is customised as it is made.

Despite the increased value of such a product, you are still limited by View's capacity to hold a printer driver of no more than 256 bytes length.

All the extra codes that one can build in, such as italic text and alternative character fonts take up memory and can only be accessed from View by temporarily redefining the second highlight code. This is very cumbersome but has always been accepted as one of View's inherent problems.

The driver generator program is very friendly, asking you for a code(s) for each facility in turn, with the options of skipping the choice or returning to its start. There are specific files provided for some common printers (the same ones as on the original printer driver cassette in fact), and loading these in presents a prompt for each value for the printer as each question is asked. This enables you to change the odd few parameters to suit, or use a provided answer file for a printer similar to your own to aid customising the program.

Once all the questions have been answered you can start all over again or save the new answer file. You may then assemble what has been produced and save the code as a new printer driver.

This package is an elegant answer to many a View user's prayers, but it highlights the deficiencies of the word processor ROM in the first place.

OUT WITH THE INKEY$

Sir Clive Sinclair has met his match in Adam Denning. Our resident machine code maestro has overridden the Spectrum's rigid single keyword commands, in a program designed to work within larger applications (screen editor . . . language compiler . . . adventure game . . .) as an interactive keyboard input.

It allows up to 255 characters to be entered into a buffer anywhere in the Spectrum's memory, with a carriage return (provided by the ENTER key) terminating the input. Deletion is available in two forms; pressing CAPS SHIFT and 0 together deletes a character in the normal way while CAPS SHIFT and 1 (EDIT) deletes the entire line entered so far.

Each key depression results in an audible beep, and CAPS SHIFT, CAPS LOCK and SYMBOL SHIFT act in the normal way. Naturally only 'real' ASCII characters can be input, so Spectrum Basic keywords and the three composite conditionals <>, <= and >= are not available. Whenever a key is depressed it appears on the screen, as you would expect, and the initial print position is easily alterable.

An extra facility offered is inverse video. Pressing CAPS SHIFT and 4 turns inverse video on, while CAPS SHIFT and 3 turns it off. Characters entered into the buffer during inverse video-on have the top bit set, but the seven bit ASCII representation remains the same.

Apart from the bit patterns of the character set, the entire routine is completely ROM independent – including screen printing. It is extremely versatile and is preferable to the Spectrum's own RST 38 keyboard scan. Before describing the routine in any more detail, a discussion of the variables and locations used will clarify the customising requirements. At present all these locations are in the printer buffer, extending from #5B00 to #5BFF, but of course these can be altered as required. The routine runs with the interrupts disabled but returns with them re-enabled. Naturally this too can be changed.

PRTPOS holds the current position. As this is initialised by the routine, any direct poking can only have adverse effects. Change the value at line 270 to alter the print position. #4000 is the top of the screen.

POSBUF holds the initial print position and is used during line deletion.

BUFPOS is used to indicate the number of characters in the buffer. Although it is allocated two bytes, only the low byte is used for counting.

BUFFER holds the start address of the input buffer – it MUST be initialised before the routine is used.

BUFEND holds the length of the buffer, which again MUST be initialised before use.

BORDER calculates and maintains the current border colour during sound generation.

INVFLG holds the current state of the video display.

CURCHR holds the cursor character, as this is changed at various times.

FLAGS1 holds various flags associated with the system.

Finally, ENDCHR defines the character code used to mark the end of input within the buffer.

When the routine is called, the necessary variables are initialised, the interrupts are disabled and a loop is entered, collecting characters and printing them or taking the requisite action. GETCHR actually scans the keyboard, looks at the shift keys and produces the code for the key or combination being pressed. It uses CHRTAB to do this, and returns special values for delete (12), delete input (7), CAPS LOCK toggle (255), and true video/inverse video switch (253 and 254 respectively).

TONAL generates a constant tone with a duration determined by the value of A on entry. The pitch is fixed.

ATTADD is entered with any screen address in HL and returns with the attribute address for that address in HL.

IMITAT and ATCTRL simulate the action of RST 10 as far as screen printing is concerned. It can only handle AT control characters, anything else below ASCII 32 or above 127 will result in undefined bit patterns being printed. If AT control codes are used then IX must point to the current character of the string being printed. IMITAT is considerably faster and shorter than RST 10, but can only handle the screen.

The rest of the routine analyses each character and reacts accordingly. Although seemingly rather long it is considerably more efficient in terms of memory than other routines the author has seen. There is room for improvement in ATCTRL, as this routine was originally written in 1982, and I can't find the latest version! Any of the individual subroutines can be freely used in user routines, but you are reminded that they are copyright of the author.

PROGRAM

```
 10 *L-
 20 CHARS  EQU   #3D00
 30 PRTPOS EQU   #5B00
 40 POSBUF EQU   #5B02
 50 BUFPOS EQU   #5B04
 60 BUFFER EQU   #5B06
 70 BUFEND EQU   #5B08
 80 BORDER EQU   #5B0A
 90 INVFLG EQU   #5B0B
100 CURCHR EQU   #5B0C
110 FLAGS1 EQU   #5B0D
120 BORDCR EQU   23624
130 ENDCHR EQU   #FF
140        ORG   55000
150        DI
160        LD    A,(BORDCR)
170        SRA   A
180        SRA   A
190        SRA   A
200        AND   7
210        LD    (BORDER),A
220        XOR   A
230        LD    (FLAGS1),A
240        CALL  TONAL
250        LD    A,#3E
260        LD    (CURCHR),A
270        LD    HL,#4000
280        LD    (PRTPOS),HL
290        CALL  GETINP
300        EI
310        RET
320 GETINP LD    HL,(PRTPOS)
330        LD    (POSBUF),HL
340        LD    HL,(BUFFER)
350        LD    (HL),ENDCHR
360        LD    DE,0
370        LD    (BUFPOS),DE
380        XOR   A
390        LD    (INVFLG),A
400 PRTLWP LD    HL,(PRTPOS)
410        PUSH  HL
420        CALL  ATTADD
430        SET   7,(HL)
440        LD    A,(CURCHR)
450        CALL  NOTATC
460        POP   HL
470        LD    (PRTPOS),HL
480        PUSH  HL
490        CALL  GETCHR
500        PUSH  AF
510        LD    A,#20
520        CALL  TONAL
```

```
 530        POP  AF
 540        CALL DELAY
 550        CP   #D
 560        JR   Z,FINISH
 570        JR   C,DELETE
 580        CP   #FD
 590        JP   NC,CAPSLK
 600        LD   E,A
 610        LD   A,(INVFLG)
 620        AND  A
 630        RRA
 640        RRA
 650        ADD  A,E
 660        LD   HL,(BUFFER)
 670        LD   DE,(BUFPOS)
 680        ADD  HL,DE
 690        LD   (HL),A
 700        AND  #7F
 710        CALL NOTATC
 720        POP  HL
 730        CALL ATTADD
 740        RES  7,(HL)
 750        LD   DE,(BUFPOS)
 760        INC  DE
 770        LD   (BUFPOS),DE
 780        LD   HL,(BUFFER)
 790        ADD  HL,DE
 800        LD   (HL),ENDCHR
 810        LD   A,(BUFEND)
 820        CP   E
 830        JR   Z,ENDBUF
 840        JR   PRTLWP
 850 FINISH POP  HL
 860        CALL ATTADD
 870        RES  7,(HL)
 880        LD   A,#20
 890        JP   NOTATC
 900 DELETE POP  HL
 910        LD   C,A
 920        LD   A,(BUFFOS)
 930        AND  A
 940        JR   Z,PRTLWP
 950        LD   A,C
 960        PUSH AF
 970        PUSH HL
 980        PUSH HL
 990        LD   B,8
1000 BLANK  LD   (HL),0
1010        INC  H
1020        DJNZ BLANK
1030        POP  HL
1040        CALL ATTADD
1050        RES  7,(HL)
1060        LD   A,#20
1070        CALL NOTATC
1080        POP  HL
1090        POP  AF
1100        CP   #C
1110        JR   C,DLTELN
1120        DEC  HL
1130        LD   A,#FF
1140        CP   L
1150        JR   NZ,DELBLK
1160        LD   A,#3F
1170        CP   H
1180        JR   NZ,BLKDEL
1190        INC  HL
1200        JR   DELBLK
1210 BLKDEL LD   A,H
1220        SUB  7
1230        LD   H,A
1240 DELBLK LD   (PRTPOS),HL
1250        LD   DE,(BUFFOS)
1260        DEC  DE
1270        LD   (BUFFOS),DE
1280        LD   HL,(BUFFER)
1290        AND  A
1300        ADC  HL,DE
1310        LD   (HL),ENDCHR
1320        JP   PRTLWP
1330 ENDBUF XOR  A
1340        CALL TONAL
```

```
1350        LD   A,#3C
1360        CALL NOTATC
1370        LD   HL,(PRTPOS)
1380        DEC  HL
1390        CALL ATTADD
1400        SET  7,(HL)
1410 OUTKEY CALL GETCHR
1420        PUSH AF
1430        XOR  A
1440        CALL TONAL
1450        POP  AF
1460        CP   #E
1470        JR   NC,OUTKEY
1480        CP   #D
1490        JR   NC,DELCUR
1500        LD   HL,(PRTPOS)
1510        DEC  HL
1520        PUSH HL
1530        JR   DELETE
1540 DELCUR LD   HL,(PRTPOS)
1550        DEC  HL
1560        LD   B,8
1570        PUSH HL
1580 CUROFF LD   (HL),0
1590        INC  H
1600        DJNZ CUROFF
1610        POP  HL
1620        CALL ATTADD
1630        RES  7,(HL)
1640        RET
1650 DLTELN LD   A,(BUFPOS)
1660        LD   B,A
1670        LD   HL,(POSBUF)
1680        LD   (PRTPOS),HL
1690        PUSH HL
1700 CLRLNE PUSH BC
1710        LD   A,#20
1720        CALL NOTATC
1730        POP  BC
1740        DJNZ CLRLNE
1750        POP  HL
1760        LD   (PRTPOS),HL
1770        JP   GETINP
1780 CAPSLK CP   #FF
1790        JR   NZ,INVIDE
1800        LD   A,(FLAGS1)
1810        BIT  7,A
1820        JR   NZ,SETLCK
1830        OR   #81
1840        LD   (FLAGS1),A
1850        LD   A,#43
1860        LD   (CURCHR),A
1870        POP  HL
1880        LD   (PRTPOS),HL
1890        JP   PRTLWP
1900 SETLCK AND  %01111110
1910        LD   (FLAGS1),A
1920        LD   A,#3E
1930        LD   (CURCHR),A
1940        POP  HL
1950        LD   (PRTPOS),HL
1960        JP   PRTLWP
1970 INVIDE CP   #FE
1980        JR   NZ,TRUVID
1990        LD   A,1
2000 BAKVID LD   (INVFLG),A
2010        POP  HL
2020        JP   PRTLWP
2030 TRUVID XOR  A
2040        JR   BAKVID
2050 IMITAT CP   #16
2060        JR   NZ,NOTATC
2070        INC  IX
2080        CALL ATCTRL
2090        LD   (PRTPOS),HL
2100        LD   A,(IX+0)
2110 NOTATC AND  A
2120        SUB  #20
2130        LD   L,A
2140        LD   H,0
2150        ADD  HL,HL
2160        ADD  HL,HL
```

```
2170        ADD  HL,HL
2180        LD   DE,CHARS
2190        ADD  HL,DE
2200        EX   DE,HL
2210        LD   HL,(PRTPOS)
2220        PUSH HL
2230        LD   B,8
2240 RSTTEN LD   A,(DE)
2250        LD   C,A
2260        LD   A,(INVFLG)
2270        AND  A
2280        JR   Z,NOTINV
2290        LD   A,C
2300        CPL
2310        LD   C,A
2320 NOTINV LD   (HL),C
2330        INC  DE
2340        INC  H
2350        DJNZ RSTTEN
2360        POP  HL
2370        INC  HL
2380        XOR  A
2390        CP   L
2400        JR   NZ,LNESME
2410        LD   A,7
2420        ADD  A,H
2430        LD   H,A
2440        LD   A,#50
2450        CP   H
2460        JR   NC,LNESME
2470        LD   HL,#4000
2480 LNESME LD   (PRTPOS),HL
2490        RET
2500 ATCTRL LD   A,(IX+0)
2510        PUSH AF
2520        LD   H,#40
2530        CP   8
2540        JR   C,COLBLK
2550        LD   H,#48
2560        CP   #10
2570        JR   C,COLBLK
2580        LD   H,#50
2590 COLBLK POP  AF
2600        AND  7
2610        INC  A
2620        LD   B,A
2630        LD   A,#E0
2640 LINFIX ADD  A,#20
2650        DJNZ LINFIX
2660        LD   L,(IX+1)
2670        ADD  A,L
2680        LD   L,A
2690        INC  IX
2700        INC  IX
2710        RET
2720 ATTADD LD   A,H
2730        AND  #18
2740        SRL  A
2750        SRL  A
2760        SRL  A
2770        ADD  A,#58
2780        LD   H,A
2790        RET
2800 TONAL  PUSH AF
2810        PUSH BC
2820        LD   BC,#00FE
2830        LD   D,A
2840 TONE   LD   A,D
2850        CALL TONDLY
2860        LD   A,(BORDER)
2870        ADD  A,#10
2880        OUT  (C),A
2890        LD   A,D
2900        CPL
2910        CALL TONDLY
2920        LD   A,(BORDER)
2930        OUT  (C),A
2940        DEC  D
2950        JR   NZ,TONE
2960        POP  BC
2970        POP  AF
2980        RET
```

```
2990 TONDLY DEC  A
3000        RET  Z
3010        JR   TONDLY
3020 DELAY  LD   B,#60
3030 LOOP1  PUSH BC
3040        LD   B,0
3050 LOOP2  DJNZ LOOP2
3060        POP  BC
3070        DJNZ LOOP1
3080        RET
3090 GETCHR CALL GETSHF
3100        CALL NOTSHF
3110        JR   Z,GETCHR
3120        LD   E,A
3130        PUSH DE
3140        LD   A,(HL)
3150        LD   HL,CHRTAB
3160        AND  A
3170        JR   Z,NOSHFT
3180        LD   B,2
3190        BIT  1,A
3200        JR   NZ,DBLSHF
3210        DEC  B
3220        AND  #81
3230        CP   #81
3240        JR   Z,DBLSHF
3250        CP   1
3260        JR   Z,DBLSHF
3270        LD   A,D
3280        CP   4
3290        JR   C,DBLSHF
3300        CP   6
3310        JR   C,NOSHFT
3320 DBLSHF LD   DE,#28
3330 SHFTAB ADD  HL,DE
3340        DJNZ SHFTAB
3350 NOSHFT POP  DE
3360        PUSH DE
3370        LD   B,D
3380        LD   D,0
3390        LD   A,#FB
3400 THEROW ADD  A,5
3410        DJNZ THEROW
3420        LD   E,A
3430        ADD  HL,DE
3440        LD   E,1
3450        POP  BC
3460        LD   A,C
3470        CP   1
3480        JR   Z,THECHR
3490        INC  E
3500        CP   2
3510        JR   Z,THECHR
3520        INC  E
3530        CP   4
3540        JR   Z,THECHR
3550        INC  E
3560        CP   8
3570        JR   Z,THECHR
3580        INC  E
3590 THECHR LD   B,E
3600        DEC  HL
3610 CHRTHE INC  HL
3620        DJNZ CHRTHE
3630        LD   A,(HL)
3640        AND  A
3650        JR   Z,GETCHR
3660        RET
3670 GETSHF LD   HL,FLAGS1
3680        LD   A,(HL)
3690        AND  %11111100
3700        LD   (HL),A
3710        LD   A,#FE
3720        IN   A,(#FE)
3730        CPL
3740        AND  1
3750        JR   Z,NOTCAP
3760        SET  0,(HL)
3770 NOTCAP LD   A,#7F
3780        IN   A,(#FE)
3790        CPL
3800        AND  2
3810        RET  Z
3820        SET  1,(HL)
3830        RET
3840 NOTSHF LD   BC,#FEFE
3850        LD   D,8
3860        IN   A,(C)
3870        CPL
3880        AND  #1E
3890        RET  NZ
3900        DEC  D
3910 KEYLAB RLC  B
3920        IN   A,(C)
3930        CPL
3940        AND  #1F
3950        RET  NZ
3960        DEC  D
3970        LD   A,1
3980        CP   D
3990        JR   NZ,KEYLAB
4000        RLC  B
4010        IN   A,(C)
4020        CPL
4030        AND  #1D
4040        RET  NZ
4050        DEC  D
4060        RET
4070 CHRTAB DEFB 32,0
4080        DEFM "mnb"
4090        DEFB 13
4100        DEFM "lkjhpoiuy098"
4110        DEFM "7612345qwert"
4120        DEFM "asdfg"
4130        DEFB 0
4140        DEFM "zxcv"
4150        DEFB 32,0
4160        DEFM "MNB"
4170        DEFB 13
4180        DEFM "LKJHPOIUY"
4190        DEFB 12,0,0,0,0,7
4200        DEFB 255,253,254,0
4210        DEFM "QWERTASDFG"
4220        DEFB 0
4230        DEFM "ZXCV"
4240        DEFB 32,0
4250        DEFB 46,44,42
4260        DEFB 13
4270        DEFM "=+-^"
4280        DEFB 34,59,127
4290        DEFB 93,91
4300        DEFM "_)('&!@#$%"
4310        DEFB 0,0,0
4320        DEFM "<>"
4330        DEFB 126,124,92
4340        DEFB 123,125,0
4350        DEFM ":`?/"
```

From palette to pixel

Liz Gregory surveys a range of micro graphic aids.

Joysticks

Joysticks are used to control cursor movement as an alternative input device to the keyboard. They take over the specific functions of each key and, usually by means of some kind of lever or rod, control information coming from the 'analogue' movements of the user to be translated by the computer into digital signals. Often hand-held devices, they are named after the control lever of an aeroplane because of their similar design and operation.

Joysticks can be used for a number of purposes, providing a more centralised area of control than the keyboard. Commonly associated with games programs, they provide a means by which the user can act speedily when faced with the complexities of arcade style graphics.

The hardware of a joystick is relatively straightforward. They often take the form of a central rod surrounded at its base by 4 microswitches which give out an appropriate bit pattern whenever a switch is closed. However this gives only 4 possible inputs and even though a finer condition may sometimes be realised, with an intermediate stage when 2 switches are closed, the 8 inputs this enables still makes for a rigid, simplistic operation. Alternatively, 2 potentiometers with a recognisable voltage are used to give the co-ordinates of any one point. This information is then converted to binary data. Such joysticks are accurate enough for graphics and design purposes.

'The hardware of a joystick is relatively straightforward'

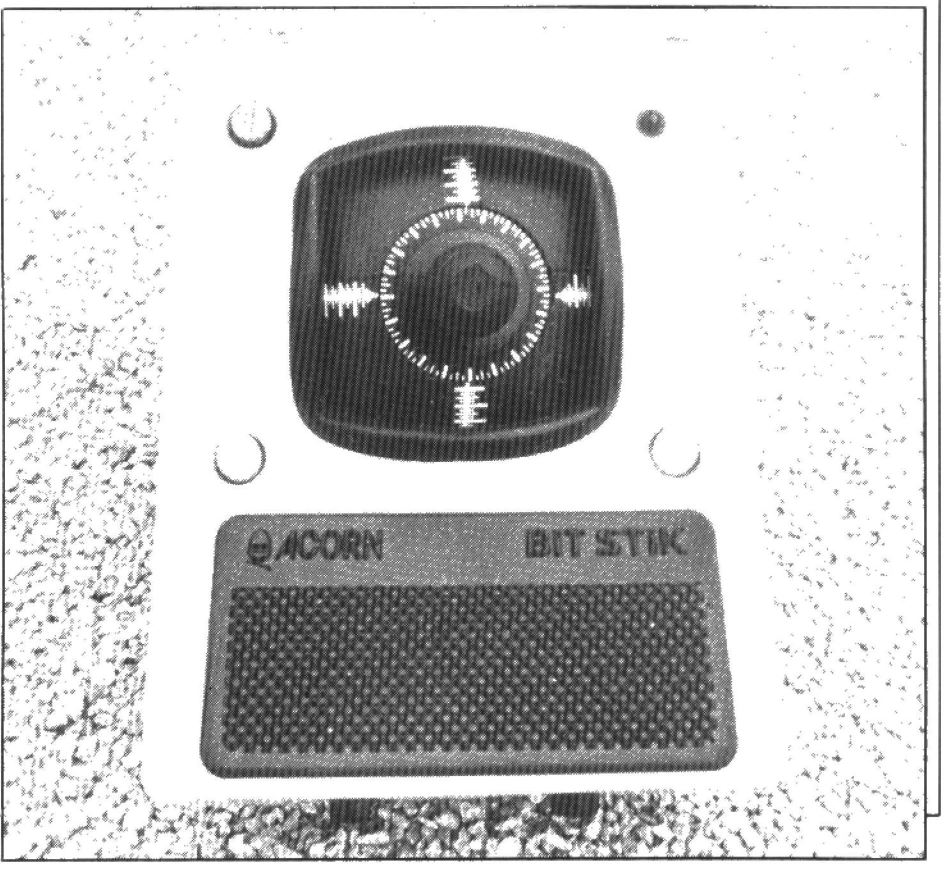
As with other peripherals, a joystick requires relevant interfacing with the machine with which it is to be used. Micros must have facilities, normally in the form of an A/D converter, whereby the joystick's movements can be translated into binary code. Few micros have a built in A/D converter (notable exceptions include the Dragon and BBC) so the first major problem is finding a suitable means of interfacing.

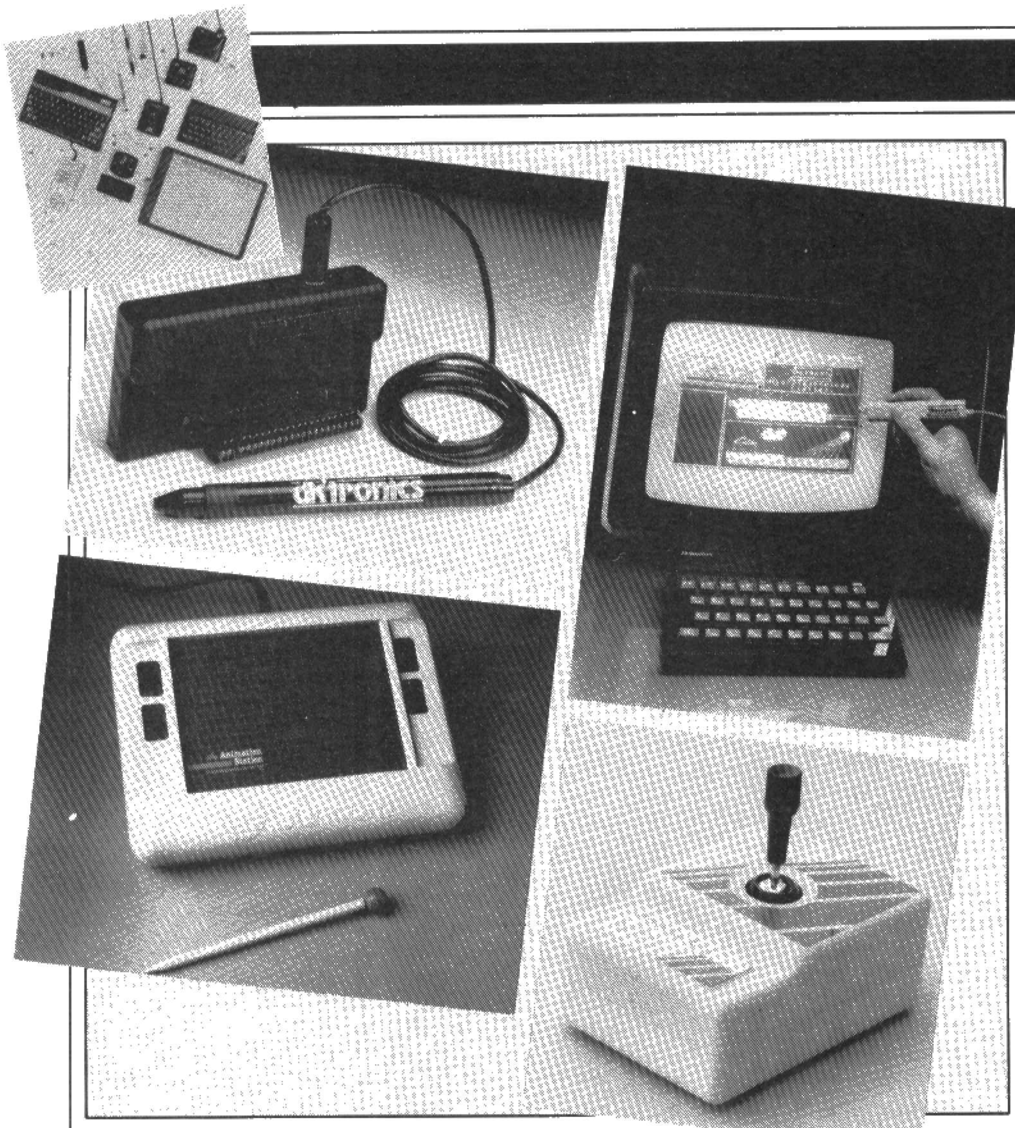
Some computers do have specific joystick ports whereby the interface can simply follow the action of the keyboard. These machines include Vic 20, Commodore 64 and the Dragon 32. However, the majority make do with a common 'user' port which can be used with an A/D converter and whereby information received from the peripheral is acted upon by the software. In fact, overcoming the difficulties of a complete lack of standardisation is by far the most frustrating task when selecting a joystick for a specific machine, bearing in mind that some software is not designed for use with peripherals, to say nothing of a myriad of pin configurations.

The problem of interfacing can be tackled from a hardware or software aspect; solutions may be found with both. It is possible to buy joysticks which may be programmed to adapt to any game whether designed for peripheral usage or not. For example, Downsway Electronics' model just plugs into the expansion bus of the Spectrum and will program games for use with the joystick. Alternatively, AGF's device is accompanied by an interface module which is twice as expensive as the joystick itself, but allows all Spectrum software to be run. Using a software solution to the problem, Cambridge Computing have an 'intelligent' joystick which comes complete with a tape which will sort out any incompatabilities. Unfortunately, all interfaces of this nature cost quite a lot.

Recently, joysticks have moved away from their traditional shape and have ceased to be simple lever-type devices. Indeed some of them are hardly recognis-

Acorn's Bitstik is one component of the most advanced graphics system available for home micros.





Pictured above are lightpens from DK Tronics and Datapen, the Koala pad and a joystick from Consumer Electronics.

able as such from their outward appearance. Devices like Suncom's Joysensor are pressure sensitive and look more like "walkie talkies" than a peripheral. The cursor may be moved around the VDU by means of a disk which is situated on the bottom of the device whilst the fire button is on the top. Light sensitive models like East London Robotics' Trickstick look more like a lightpen than a joystick and have a confusion of buttons which are sensitive to finger pressure. The most advanced and therefore most expensive model on the market is Acorn's Bitstick which can be used as a graphics tool to allow the user to shrink and enlarge pictures and even rotate and zoom in on detail within them. It is very costly but is really intended for more complex computer aided design work.

Novel ideas are obviously aimed at overcoming an overcrowded market but there are some decent modifications upon traditional joystick shapes which suggest that change may not be completely unnecessary. For example, the Quickshot joystick has suckers which will stick fast to flat surfaces and thus enable one-handed operation with relative ease. This should prove advantageous for avid games players. Some of the joysticks around are so cheap and nasty that they will not stand up to much useage. It is worthwhile experimenting with different ones so that robustness in the face of over-excitement is a proven factor before purchase.

Lightpens

Lightpens are used in conjunction with a VDU. Interaction between the screen, lightpen and computer allows the user to draw pictures directly onto the screen. Depending upon software effectiveness, they are usually associated with enhancing a micro's graphics capabilities and can be a major asset in design work. They may also be used for games purposes but are more suitable for 'thinking' style games like Chess, rather than speed and skill arcade tests.

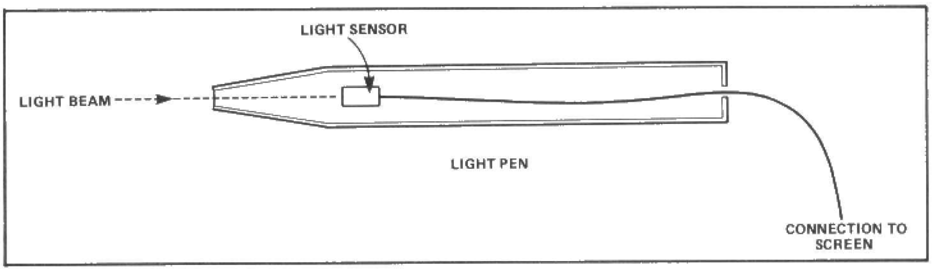


Figure 1 shows the sensor situated in the tip of the pen.

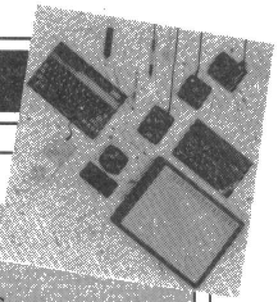
Lightpens are designed to enable the computer to locate the moving dot of light which indicates the presence of the electron beam on a cathode ray tube display. Each frame which appears on the VDU is made up of moving rather than static matter and, with its light sensitive tip, the lightpen can pick up the position of the dot when held close to the screen. Information about any character which may be present upon the VDU is sent back to the computer via a photo-transistor. The software then converts this information into binary data, to illuminate the pixels which the pen has passed over, thus drawing upon the screen (see **Figure 1**).

Again, lightpens require interfacing and again, some computers have specific facilities to accommodate them. For example, the BBC machine has a dedicated device in the form of a 6845 Cathode Ray Tube Controller which with a light pen register allows the machine to calculate the position of the pen. This special chip uses the scanning line of the VDU to inform the software of the pen's whereabouts. Thus positioning is much more accurate and the drawing is likely to have a clearer and more accurate definition.

Graphics tablets

Graphics tablets transfer pictorial matter, whether it be drawing or diagrams, from a sheet of paper instantly onto the screen. A relatively new addition to the peripheral front, they save enormously on the labour and complexities involved in reproducing graphics on a computer which normally involves an awful lot of planning and plotting co-ordinates and then writing the program. With a graphics tablet, the image you want whether by drawing free hand or tracing, can be drawn, stored and reproduced by other peripherals such as printer, plotter or screen as required. Again these tablets work with extensive software which is written in machine code to speed up the drawing process.

Graphics pads are of two types. There is a simple tracing device which involves the copying of a picture via a pantograph arm. This is obviously adequate for straightforward work but any unusual shapes are out of the question. A more accurate but much more expensive device is a graphics digitising tablet. This takes the form of a flat surface plate, made up of a grid of wires which are encompassed in a base. When touched by a stylus, the wires are brought into electrical contact and information is sent to the computer which determines the co-ordinates of the touch. The computer is thus able to locate the position of the pen and treats this touch as a form of input. By conversion into a digital form, at this point, the tablet is able to have each part of the picture put onto the computer. It is an expensive form of graphic reproduction



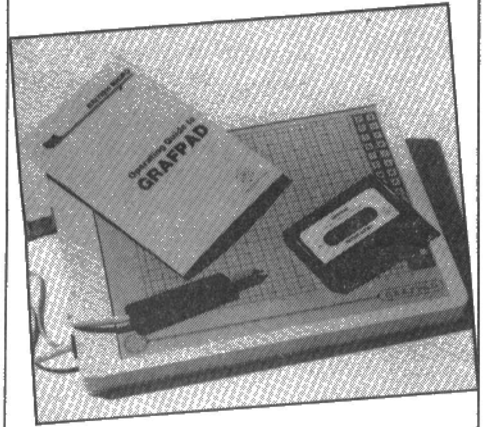
Peripheral Type	Name	Price	Machine/Interface	Company
Light Pen		£19.95	CBM	Tremvier
Light Pen	Datapen	£25.00	BBC, Dragon, CBM64	Datapen
Light Pen		£28.75	CBM64, BBC	Stack
Light Pen		£12.95	BBC	Minor Miracles
Light Pen		£18.00	BBC	Watford Electronics
Light Pen		£19.95	Spectrum	DK Tronics
Light Rifle		£29.95	CBM64, BBC	Stack
Light Pen		£45.95	BBC	RH Electronics
Light Pen	Pixstick	£29.95	CBM64	Dams
Joystick	Quickshot	£9.95	CBM64	VPD
Joystick	Quickshot	£11.95	CBM64	VPD
Joystick	Quickshot	£22.95	Spectrum	AGF
Joystick	Comp Pro	£10.99	CBM64	Kempston
Joystick	Comp Pro 3,000	£12.75	CBM64	Kempston
Joystick	Kraft	£13.75	CBM64	SPD
Joystick	Joysenser	£29.95	CBM64	Consumer Electronics
Joystick	Sureshot	£15.95	CBM64	Cookridge Computer Supplies
Joystick		£8.95	BBC	Flight Link
Joystick		£24.95	BBC	Voltmace
Joystick		£22.95	Spectrum	Downsway
Joystick	Spectrum Stick	£9.95	Spectrum	Grant Design
Joystick	Trickstick	£34.50	Spectrum	E London Robotics
Joystick	Le Stick	£24.95	Spectrum	Datasoft
Joystick	Trackball	£39.99	Atom/CBM64	Atari
Joystick		£29.90	Spectrum	Cambridge Computing
Graphics Tablets	Grafpad	£143.75	BBC, Spectrum	British Micro
Graphics Tablets	Touchmaster	£99.00	Dragon	GEC
Graphics Tablets	Koala pad	£80.00	CBM64	Audiogenic
Graphics Tablets	Animation Station	£80.00	CBM64	Consumer Electronics

but a very attractive and accurate one.

There are not too many graphics tablets on the market but the indications are that this will change fairly rapidly. Of the models available, British Micro make Grafpad for both the Spectrum and BBC machines and which retail at £143.75 (inc. VAT) and are thus fairly expensive. Both the BBC and Spectrum versions have extensive software included.

Audiogenic distribute the Koala pad for the Commodore 64 and CIC 20 at around £80. The Koala pad has a wide range of facilities which include colour transfer, copying and filling-in options as well as the possibilities of zooming in. It is very attractively packaged.

The indications are that rivals to these tablets will soon be arriving and for the Commodore market alone, Personal Peripherals have only to find a UK distributor for their Sketch Pad which will retail at £59, before they will start importing their tablets. Sketch Pad is capable of reproducing even plaid patterns and is very simple to operate which must have an appeal for the educational market. Aimed at home and business users, the substantial software support comes in ROM form and there are hopes that it may be interfaced with all machines. This is an aspiration shared by GEC with their Touchmaster which is due to be launched officially in September of this year. It will retail at £99 and although relatively expensive, it looks an extremely attractive add-on.



British Micro's Grafpad is available in three versions designed for use with the Spectrum, BBC micro and Commodore 64. The 'pad, together with comprehensive cassette based software allows complex graphic displays to be produced by way of a wide range of different drawing modes. The Grafpad is well built and offers a means of producing computer generated images.

Names and addresses

Quickshot
VPD
GEC Industrial Estate
Wembley
Middlesex

Consumer Electronics
Failsworth
Manchester MS5 0HS

Sure Shot
Cookridge Computer Supplies
PO Box IW9
Leeds LS1 6NT

AGF
26 Van Gogh Place
Bognor Regis
W Sussex PO22 9BY

Downsway Electronics
Depot Road
Epsom
Surrey KT17 4RJ

East London Robotics
Gate 11
Royal Albert Dock
London E16

Cambridge Computing
1 Benson Street
Cambridge CB4 3QO

Minor Miracles
PO Box 48
Ipswich IP4 1AR

Stack Computer Services
290-295 Derby Road
Bootle
Merseyside

DK Tronics
Unit 6
Shire Hill
Saffron Walden
Essex CB11 0AQ

RH Electronics
Chesterton Mill
French's Road
Cambridge CB4 3NP

British Micro
Unit Q2
Penfold Works
Imperial Way
Watford
Herts WD2 4YY

Personal Peripherals Inc
980 North Beltline Road
Suite 20
Irving
Texas 73061 USA

Killing Bugs

In the 'Killing Bugs' article on Spectrum hardware modifications (June 1984 *E&CM*) two diagrams were unfortunately omitted. The diagram simply explains the introduction of an extra inverter between MREQ and pin 27 of the new ROM replacement.

Also the tables below represent a significant improvement in the ease of use of the NMI break routine. This can be achieved if the address at which the RETURN TO BASIC (C9) occurs, is held in a system variable to which the operator has access. In the original article this 7530H ie 30,000 address was fixed in EPROM and required that the M/C development area always begins at this position within the memory. The modified program allows the operator to POKE 23296 and 23297 with the address where the RETURN TO BASIC (C9) is required. This technique should allow almost any program crash to be broken and control regained.

TABLE 2. Basic start to modified monitor program

```
10  POKE 23296, 48    address of
20  POKE 23297, 117   C9 introduction
30  PRINT USR 15424
40  STOP
```

TABLE 3. NMI vector addressing

```
3C40   3E50       LDA,50
   2   32 B0 5C   LD(5C B0)A
   5   3E 3C      LDA 3C
   7   32 B1 5C   LD(5C B1),A
   A   C3 00 3C   JP 3C00  →    To original
                                start of
                                monitor
                                Page 79
                                E&CM
                                June '83
```

TABLE 4. The return to BASIC called by NMI

```
3C50   ED6B005B   LDHL,5B00
   4   36C9       LD(HL),C9
   6   C3 70 00   JP 0070
                     ↓
             Return to Spectrum ROM after having
             loaded C9 into address found at
             5B00 + 5B01 (23296 + 23297)
```

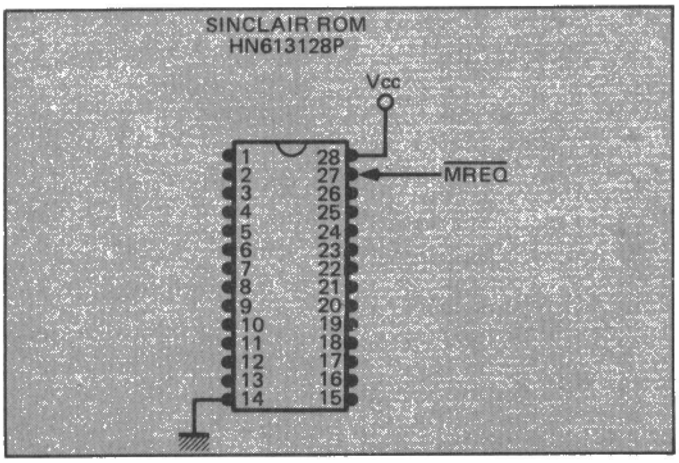


Figure 1. Sinclair ROM HN613128P.

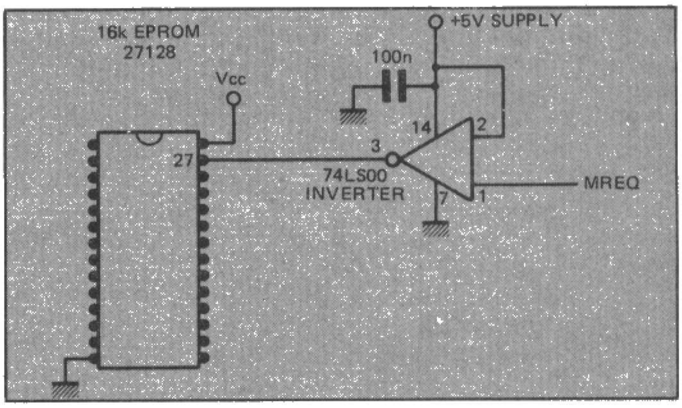


Figure 2. Details of modified circuit to use EPROM.

Centronics Printer Buffer

It has come to my notice that there are few errors in the circuit diagram for the Centronics Printer Buffer, published May 1984.

Firstly, IC5 is listed as a 74LS06, this should be 74LS04.

The other errors concern pin connections to the 8255 (IC3).

The RESET pin is un-numbered and should be pin 35. The pin marked AD0 should be pin 8. The pin marked AD1 should be pin 9.

I hope that this will help any readers who have had any problems with the circuit.

Robert Harvey

Added to P.38 of ⑥ [illegible] MAY 1984

Spectrum Trace

With reference to my article kindly published in your April, 1984 issue (Spectrum Trace page 20), myself and some of your readers have pointed out the following errors:

The hex loader:

```
130  IF INKEY$="" THEN GO TO 130
220  LET a$=a$+b$
```

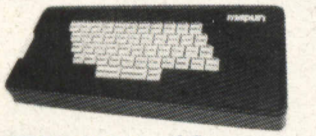
Also you omitted to state that the start of the machine code is 65000 (FDE8 hex) and that any numbers included in the hex-dump between brackets must NOT be typed in. Also to save the code type: SAVE "trace" CODE 65000,215.

Since some readers have complained to me about the style of the hex-dump, I include below a new hex-dump which you may like to publish to please your readers! your readers!

Nicholas F. Ryman-Tubb

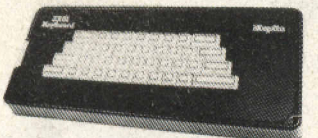
```
FDE8    21 F7 FD 22 FF FE F3 3E FE ED 47 ED 5E FB C9 FF
FDF8    F5 E5 21 B4 FE CD 89 FE B7 28 6C 3C CA A9 FE 2A
FE08    45 5C 24 28 62 25 C5 D5 E5 ED 5B C0 FE ED 52 E1
FE18    28 53 22 C0 FE E5 21 BA FE CD 89 FE B7 28 09 47
FE28    C5 06 00 10 FE C1 10 FB E1 AF 47 4F 11 E8 03 ED
FE38    52 3C 30 FB 19 3D E5 CD 72 FE E1 11 64 00 AF ED
FE48    52 3C 30 FB 19 3D 0E 01 E5 CD 72 FE E1 11 0A 00
FE58    7D 93 14 30 FC 15 83 67 E5 0E 02 7A CD 72 FE F1
FE68    0E 03 CD 72 FE D1 C1 E1 F1 C9 21 1C 40 09 87 87
FE78    87 EB 4F 21 80 3D 09 06 08 7E 2F 12 23 14 10 F9
FE88    C9 E5 2A 5D 5C E3 22 5D 5C D5 C5 DD E5 CD B2 28
FE98    DD E1 C1 D1 3E 00 38 04 23 23 23 7E E1 22 D5 5C
FEA8    C9 F3 3E 3F ED 47 ED 56 E1 F1 FB C9 74 72 61 63
FEB8    65 0D 73 70 65 65 64 0D 00 00 00 00 00 00 00 00
```

EXCITING ADDITIONS FOR YOUR HOME COMPUTER



KEYBOARD with ELECTRONICS for ZX SPECTRUM

★ Full size, full travel keyboard that simply plugs into expansion port on your Spectrum. ★ Offers single key selection of all major multi-key functions. ★ Extends port for other peripherals. ★ Can accept Atari-type joysticks (optional extra — order 2 of FG66W, £1.36 each and note that case will require cutting).
Three kits needed to build unit: Order LK29G, LK30H & XG35Q. Total price £39.95. Full construction details in Project Book 9 XA09K 70p. Also available ready-built. Order As XG36P. Price £44.95.



KEYBOARD with ELECTRONICS for ZX81

★ Full size, full travel keyboard that's easy to add to your ZX81. ★ No soldering in ZX81; simple instructions make it easy to fit. ★ Makes Shift Lock, Function & Graphics 2 single key selections.
Complete kit (excl. case) LW72P Price £23.95. Case XG17T £4.95. Full construction details in Project Book 3 XA03D. Price 70p. Ready-built in case XG22Y. Price £32.50.

MODEM

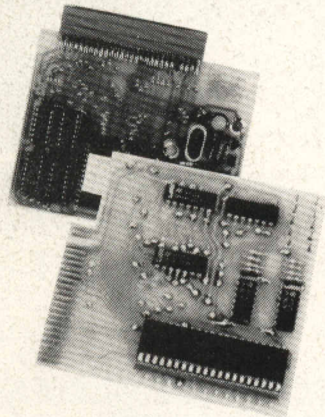
A CCITT standard modem that connects directly to your telephone line via a BT approved transformer. Transmits and receives simultaneously on European standard frequencies at 300 baud. May be used to talk to any other 300 baud European standard modem including the Maplin Computer Shopping modem on 0702 552941 and any British Telecom Datel 200/300 Service modem. The modem's computer interface is RS232 compatible.
Complete kit (excl. case) LW99H. Price £44.95. Case YK62S £9.95. Full construction details in Project Book 5 XA05F Price 70p.

INTERFACES for MODEM

Interfaces are now available for the following machines: Commodore 64, Dragon 32, Oric, Spectrum, VIC20 and ZX81. Each is complete with a Machine Code Communications program. The BBC micro needs no interface and a suitable program is on Maplin catalogue page 15 or Project Book 8, page 59.

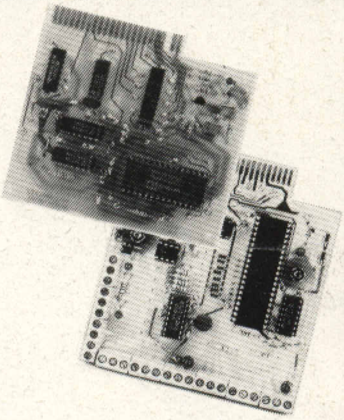
Computer	Order	Details	Price
64/VIC20	LK11M	Book 7	£9.45
Dragon 32	LK12N	Book 8	£14.95
Oric 1	LK40T	Book 10	£13.95
Spectrum	LK21X	Book 8	£19.95
ZX81	LK08J	Book 7	£29.95

**Project Book 7 XA07H. Price 70p.
Project Book 8 XA08J. Price 70p.
Project Book 10. XA10L. Price 70p.**



ZX81 I/O PORT

★ Provides two bi-directional ports for 16 input or output lines. ★ One buffered output which can interface directly to CMOS. ★ On board address selection permits expansion to six ports with two boards.
Complete kit LW76H. Price £10.49. Full construction details in Project Book 4 XA04E. Price 70p.



MAPLIN TALK-BACK SPEECH SYNTHESISERS

★ Unlimited vocabulary with allophone (extended phoneme) system. ★ Can be used with unexpanded Oric 1, VIC20 or ZX81 as it does not require large areas of memory. ★ Speech may be easily added to programs. ★In VIC20 version speech output is direct to TV speaker with no additional amplification needed.

Computer	Order	Details	Price
Oric 1	LK28F	Book 9	£23.95
VIC20	LK00A	Book 6	£22.95
ZX81	LK01B	Book 6	£19.95

**Project Book 6 XA06G. Price 70p.
Project Book 9 XA09K. Price 70p.**

DRAGON 32 I/O PORT

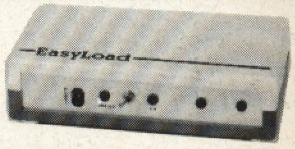
★ Provides two TTL & 3-state bus compatible 8-bit ports, ★ Four norm/inv. latched ports, ★ Two relay switched ports ★ And two opto switched ports. ★ Module plugs directly into cartridge socket and is fully programmable from BASIC.
Complete kit LK18U. Price £14.95. Full construction details in Project Book 8 XA08J. Price 70p.

MAPLIN CATALOGUE

Full details of all Maplin's projects and electronic components in our huge 480 page catalogue. On sale now in all branches of W.H. Smith price £1.35. Or send £1.65 (incl. p&p) to our mail order address.

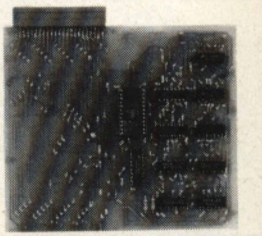
OTHER PROJECTS

For full details of our other computer-related projects please see the relevant project book as below:
**BBC Motherboard — Book 11.
ZX81 Sound on TV — Book 6.
ZX81 Extendi-RAM — Book 9.
VIC20 Extendiboard — Book 9.
Dragon 32 Extendiport — Book 10.
TTL/RS232 Interface — Book 9.
Project Book 6 XA06G. Price 70p.
Project Book 9 XA09K. Price 70p.
Project Book 10 XA10L. Price 70p.
Project Book 11 XA11M. Price 70p.**



SPECTRUM EASYLOAD

★ Greatly reduces cassette LOADing & SAVEing problems on Spectrum. ★ Battery powered, no bus connections. ★ Charging from Spectrum PSU. ★ SAVE & LOAD indicators.
Complete kit (excl case) LK39N. Price £9.95. Full construction details in Project Book 10 XA10L. Price 70p.

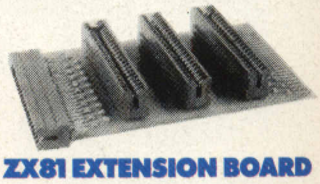


ZX81 HI-RES GRAPHICS

★ Full 256 x 192 fine pixel display with normal or inverted video. ★ Draws lines, circles and triangles, fills and textures. ★ Up to 32 user defined graphics. ★ Operates directly from extended BASIC.
Complete kit LK23A. Price £27.50. Full construction details in Project Book 9 XA09K. Price 70p.

ZX81 SOUNDS GENERATOR

★ Turns your ZX81 into a mini-synthesiser. ★ 3 programmable tone generators. ★ 3 programmable attenuators. ★ Noise generator with 3 pitch levels for special effect sounds. ★ Single address access with PEEK & POKE. ★ Connects directly to extension board or expansion port socket with extra socket (order RK35Q £2.20) ★ Requires separate amp and speaker.
Complete kit LW96E. Price £13.49. Full construction details in Project Book 5 XA05F. Price 70p.



ZX81 EXTENSION BOARD

★Plugs directly into ZX81 expansion port. ★ Accepts a 16K RAM pack and three other plug-in modules simultaneously. Parts are sold separately as follows:
PCB GB08J. Price £2.40. Edge Connectors (4 needed) RK35Q. Price £2.20 each. Track pins (1 pack needed) FL82D. Price 85p per pack of 50.

MAPLIN
ELECTRONIC SUPPLIES LTD

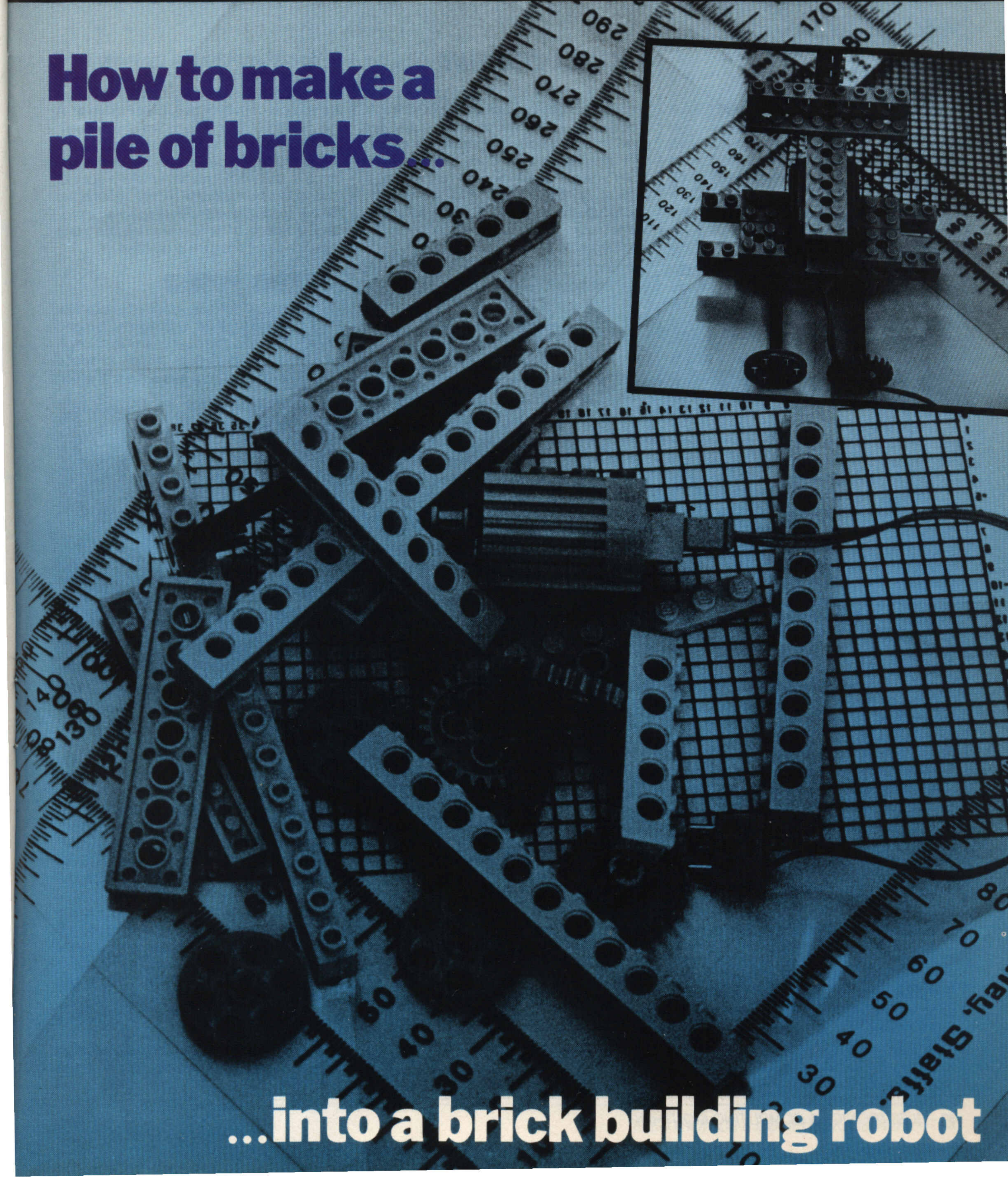
Maplin Electronic Supplies Ltd. Mail Order: P.O. Box 3, Rayleigh, Essex SS6 8LR.
Tel: Southend (0702) 552911. • Shops at: 159-161 King Street, Hammersmith, London W6. Tel: 01-748-0926.
• 8 Oxford Road, Manchester. Tel: 061-236-0281. • Lynton Square, Perry Bar, Birmingham. Tel: 021-356-7292.
• 282-284 London Road, Westcliff-on-Sea, Essex. Tel: 0702 554000. • 46-48 Bevois Valley Road, Southampton.
Tel:0703 25831. All shops closed all day Monday.
All prices include VAT and carriage. Please add 50p handling charge to orders under £5 total (except catalogue).

your ROBOT

An EMAP Publication

BRITAIN'S FIRST ROBOTICS MAGAZINE AUGUST 1984

How to make a pile of bricks...



...into a brick building robot

DIY ROBOT WALL BUILDER

Richard Sargent's do-it-yourself robot explodes the myth that robots are 1) complicated, 2) expensive, and 3) cute!

®LEGO is a registered trade mark

The first of our LEGO robots is a simple 2-axis design which is capable of moving light polystyrene blocks from A to B in a manner which implies that it possesses a fair amount of machine "intelligence". The blocks are arranged as a pyramid-shaped wall, and the robot dismantles the blocks and rebuilds them some way away, in the style of the Towers of Hanoi game. **Figure 1** shows the blocks in their old and new positions. This particular robot has no intelligence as such – it merely moves along pathways predefined in a program written on the host computer – but it is as good an introduction to machine control as any other model might provide.

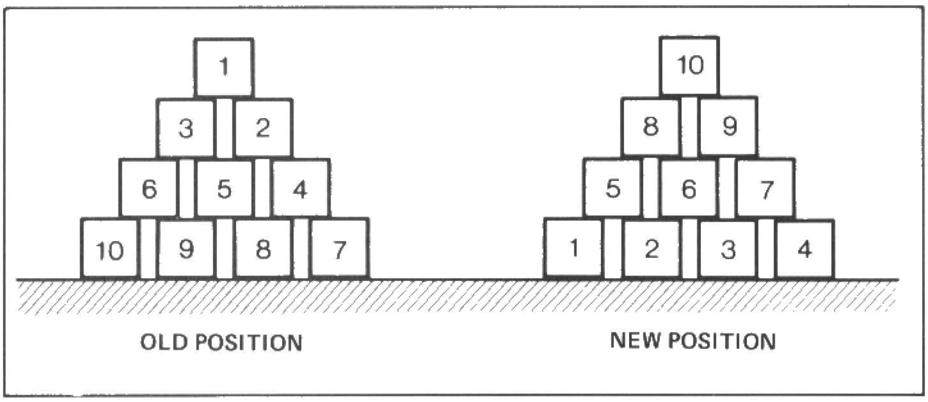


Figure 1 depicts the blocks in their old and new positions.

If you haven't used the LEGO system at all, or at least not in serious(?) applications, you will perhaps be surprised to discover what an extremely accurate building medium it is. If you are a sceptic, and have memories or actual experience of hundreds of tiny pieces of plastic lying all over the living-room carpet because a certain model "came apart" then you are in for a pleasant surprise – short of dropping them from a first-floor window, the **Your Robot** models don't fall apart.

WALL BUILDER is our smallest Robotic Thing and uses very few parts. Normally, only the critical parts of a model will be described and named, but this one has so few parts that they are all listed in the LEGO components list.

Figure 2 shows the overall model. Two 4.5V geared LEGO motors are used, stacked one on top of the other. The lower motor provides movement along the X axis and the Upper motor provides movement along the Y axis. The range of movement on the prototype is 80cm (X axis) and 25cm (Y axis), but the Y axis could be reduced to 12cm to conserve materials and the robot would still be a viable proposition.

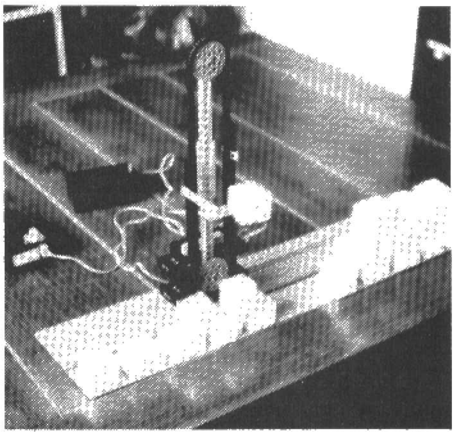
VERTICAL THINKING

Precise X-axis positioning is achieved by running the robot on standard LEGO track. Wooden or plastic rail glued to a wooden base would be equally suitable if the track isn't available. The Y-axis uses LEGO chain in an endless loop, which acts as a vertical conveyor belt and moves the blocks up and down. The prototype uses a packet of No. 1230 broad chain, lengthened with the addition of No. 1227 narrow chairn, but any combination of broad and narrow chain will do the job and the total length can be considerably less than the present total of 94 links. Chains always run smoothly when they pass over the largest of the LEGO gears and we used the two big gears from the No. 1227 packet, but the medium and medium-large gears do work so they may be used instead. The lower gear is driven directly by an axle passing through the Y-axis motor and this axle also passes through the bottom hole of a 15-hole technical beam. This beam is held upright by pegging it to a 5-hole beam which clips to the top of the Y-motor housing. You may want to limit the height of your Wall Builder to the length of the 15-hole technical beam, but we used two such beams on the prototype so that we could build high walls! The blocks in the walls are "captured" by the simple ploy of skewering them with a projection held on the vertical conveyor belt. (See **Figure 3**). A thin 8-stud LEGO plate holds the skewer. This type of plate can clip onto any single link of the broad chain, and this is the one place in the whole model where a spot of glue would be welcome – even though the blocks are lightweight, leverage over a prolonged space of time will prise the plate from the link if the glue is omitted.

> **'WALL BUILDER is our smallest robotic thing and uses very few parts . . .'**

BATTERY POWER

The motors, which are supplied in box No. 107, come complete with axles, flanged wheels, and a battery compartment which takes the three C-cells which produce 4.5V. The battery box has a switch which provides the OFF/FORWARD/REVERSE control and the model may be tested as soon as it is built. The price of batteries being what they are, it seems a good idea to use NiCad rechargeable cells for any motor-driving application. However there are two important things you should know about NiCad batteries before you start to use them. Firstly, they produce 1.2V per cell rather than the nominal 1.5V produced by other C-cell types. Running a version of Wall Builder using a 3.6V NiCad pack will require marginally different software timing, since you will be driving the motors slightly underpowered. Alternatively, an extra cell may be added to give a 4.8V supply. This has the advantage of being able to run the TTL and CMOS chips, but unfortunately the LEGO battery holder doesn't cater for four battery cells, so you would need a different one. Battery power for the chips is a particularly neat solution to the problem of obtaining +5V from the computer. Some Centronics ports have



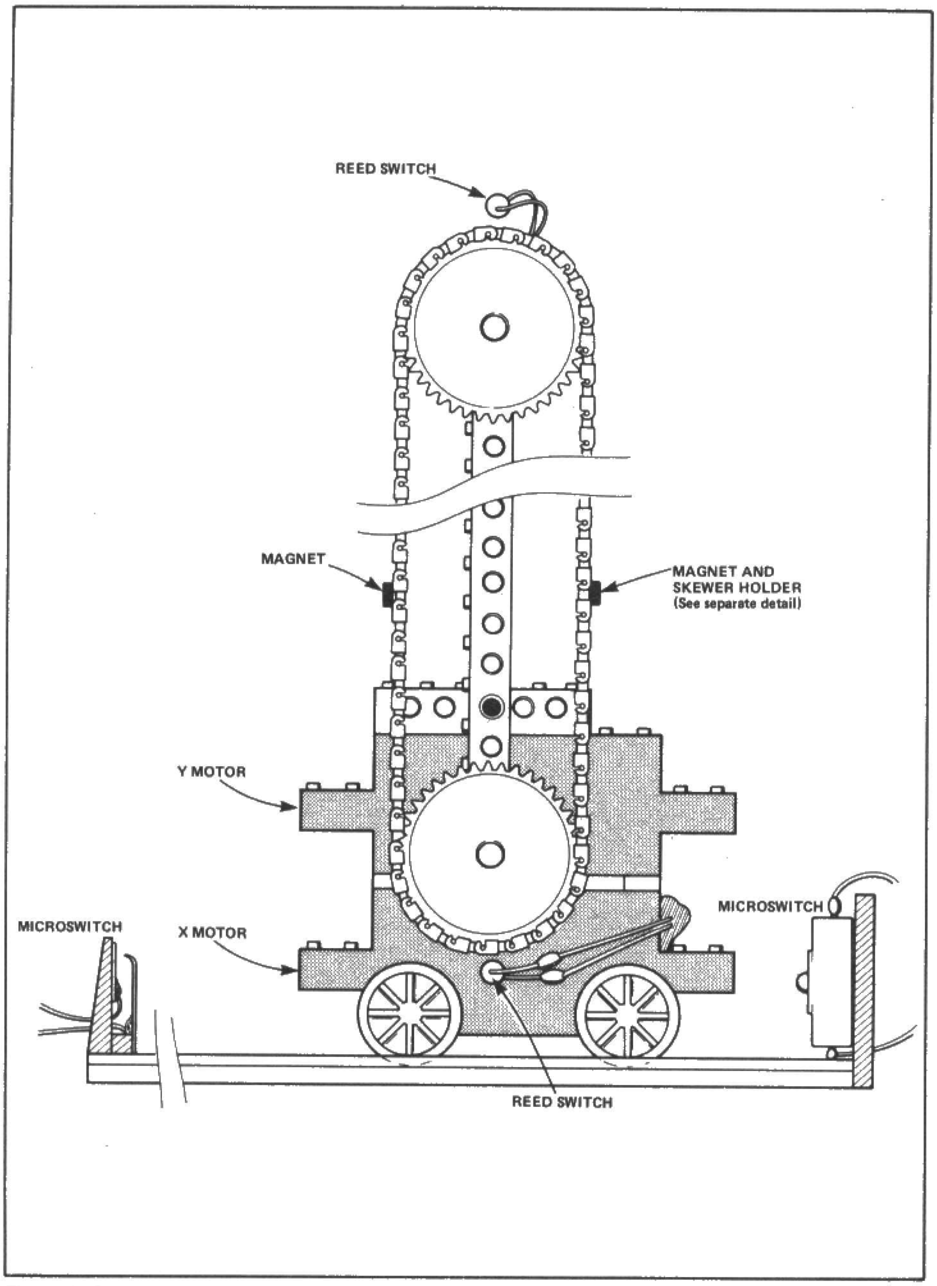


Figure 2. Overall view of the brick building robot.

+5V on pin 18, but that is by no means always the case. A second consideration when using NiCads is that they should not be short-circuited. They can deliver a large current in a short space of time causing unpleasant side effects such as burnt-out wiring. Some of the more recent LEGO battery-boxes have thermal cut-outs fitted which offer protection against this particular feature of rechargeable batteries.

The chain-loop moves at about 50cm/sec while the robot trundles along at about 20cm/sec. At this speed there is the problem of inertia when the power is switched off, and while the wall can be quite easily dismantled at this speed (!!) it cannot be rebuilt. Slower speeds and greater accuracy are required, and this is provided for by the controlling software.

THE MOTOR INTERFACE

The interface electronics for the robots have been designed to run from a Centronics or similar latched-output port. This has been done in an attempt to untie the projects from any particular computer. On machines which have no Centronics port, you are advised to complain to the manufacturer that the exclusion of this most useful facility is short-sighted, incomprehensible, wicked and so on. Having thus wasted a 16p postage stamp you should then buy yourself a Z80 PIO or a 6521 PIA and set to work building a small circuit which can emulate a Centronics port. **Figure 4** shows such a circuit for the Sinclair Spectrum. The circuit is simple, but you must remember to include software routines to configure the insides of the chip, which are quite complicated, (see **Figure 4** and **Table 1**).

SET CONTROL ON "A" LINES	PORT FDDF 64991
SETY CONTROL ON "B" LINES	PORT FFDF 65503
R/W DATA TO "A"	PORT FCDF 64735
R/W DATA TO "B"	PORT FEDF 65247

OUT 64991,207:OUT 64991,253:
OUT 65503,15 Sets the PIO to a Centronics specification
The "paddr" for OUT is 65247
The "paddr" for IN is 64735

TABLE 1

Wall Builder requires the circuit shown in **Figure 5.** It comprises two identical motor-driving circuits built around IC1, IC2 and the power transistors. IC3 allows the reading of up to eight sensors, but only four are fitted on Wall Builder. The TIP32 transistors are made to conduct by placing a logical LOW on their base pins, while the TIP31s conduct when a HIGH is placed on their base pins. Therefore, to drive the motor M1 in one direction transistors Q1 and Q2 are driven fully ON while transistors Q3 and Q4 are kept fully OFF. To reverse direction Q1 and Q2 are turned OFF while Q3 and Q4 are turned ON. The base of these transistors can be driven from standard LS series TTL and it follows that each base **could** be driven directly from the Centronics interface. It is not recommended that this be done. If mistakes are made in the wiring it is preferable that a TTL gate on the interface board should blow rather than the computer's 40-pin I/O device which will, of course, be unsocketed and in a place on the PCB where you couldn't change it anyway. In fact ICs 1 and 2 are not merely buffers. Their gates detect the DIRECTION Bit and ensure that all four transistors are not turned on at once. If the transistors were to be driven **directly** from a 4-bit motor ON/OFF pattern then faulty software or a system crash could put out the very bit-pattern which would burn your TIP31/2s to

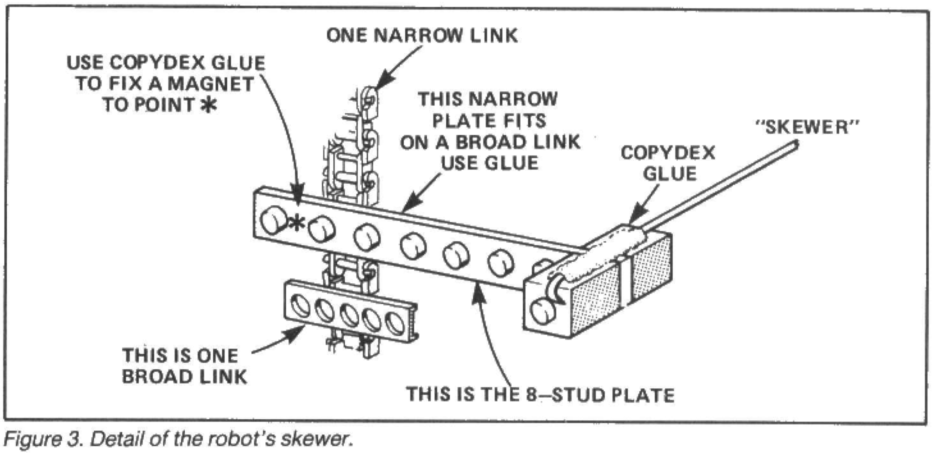

ONE NARROW LINK
USE COPYDEX GLUE TO FIX A MAGNET TO POINT ✱
THIS NARROW PLATE FITS ON A BROAD LINK USE GLUE
"SKEWER"
COPYDEX GLUE
THIS IS ONE BROAD LINK
THIS IS THE 8–STUD PLATE

Figure 3. Detail of the robot's skewer.

a frazzle. In the Wall Builder software, BIT0 represents Motor1 ON/OFF, and BIT1 represents Motor1 FORWARD/REVERSE. BITS 2 & 3 control Motor2 in a similar manner. Further transistor protection is provided by diodes D1-D4 which provide a path for the damaging back-EMF generated by the motor during the few microseconds it takes to stop revolving.

BITS 4, 5, 6 cause IC3 to select one of eight sense inputs from devices fitted to the model. The CMOS selector, type 4051 is used in preference to the TTL 74LS151 since the 4051 consists of bilateral switches which will allow **analogue** information to pass.

When handling CMOS ICs it is best to bear a few basic safety rules in mind. While modern devices with a B suffix, eg 4016B have a diode protection network on their input that make them far less prone to damage by static electricity than early devices, it is best to keep the ICs in the silver foil or conductive tube in which they (should) have been supplied until they are to be inserted in the circuit. Before handling the ICs it is also wise to touch some metal surface that is connected to ground. If you live in the type of house where you are continually being 'zapped' by static electricity you should take particular note of these words of wisdom.

The Wall Building Robot uses IC3 to read standard TTL voltage levels, and the information is routed to the READ BIT of the Centronics Port, called BUSY or ACK. In a different application a voltage from a sensor might be read and routed to an analogue-to-digital converter rather than to the Centronics Port.

Resistors R9, 10 & 11 are required to assist the Centronics Port in driving the CMOS chip IC3, while resistor R12 holds the BUSY line high. Whenever a sensor is triggered, the BUSY line will go to ground potential.

SOFTWARE

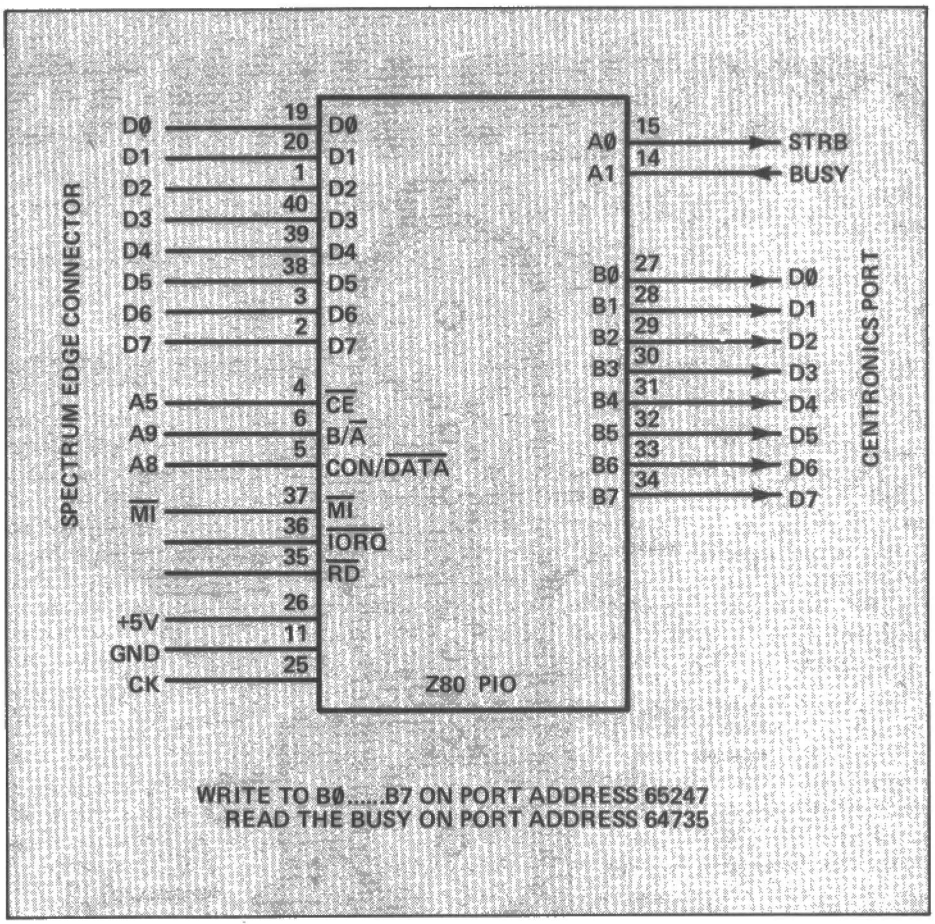
There are four levels of complexity possible for Wall Builder's software. Let's deal with the simplest level first.

If the ADDRESS of your computer's Centronics port is not known to you then you will have to communicate with your robot using BASIC LPRINT statements. In this case you must tie the Centronics BUSY/ACK line to ground, and it will not be possible to read the robot's sensors. A LOW on BUSY/ACK is a pre-requisite for LPRINT to function.

The control byte sent to LPRINT is calculated as follows:-

Shut down	00000000	0
Motor1 on, forward	00000001	1
Motor1 on, reverse	00000011	3
Motor2 on, forward	00000100	4
Motor2 on, reverse	00001100	12

'a satisfactory relationship may be established so that the robot's axis may be moved by the height of one block'



WRITE TO B0.....B7 ON PORT ADDRESS 65247
READ THE BUSY ON PORT ADDRESS 64735

Figure 4. Minimum configuration Spectrum centronics port.

A motor driving subroutine will take the form:-

```
  10  LET V=20:REM set V by
      experiment
  20  LET M=1:REM motor1, forward
  25  LET C=0:REM bit patter for
      bits 4, 5, 6
  30  GOSUB 1000
  40  STOP
 999  REM move motor M a few
      millimetres
1000  LPRINT M+C:REM MOTOR ON
1002  FOR D=1 TO V: NEXT D:REM
      DELAY
1004  LPRINT 0:RETURN:REM OFF
```

Without being able to sense its position, the Wall Builder will gradually become less accurate, with the software thinking it is in one position when in reality it is in another. It is surprising how long it can maintain reasonable accuracy, especially if slow speeds are used. There is no slip in the gearing and the slack which exists when a motor is instructed to reverse can be taken into account by the controlling program.

In the listing, line 1002 is a delay loop which forms the +5V "plateau" of the motor's ON pulse. It should be set so that a single pulse runs the LEGO motor for about 0.1 sec. A pulse shorter than this tends to cause the motor to run in a rather unreliable manner. To build up a movement of a few centimetres the subroutine 1000 is called by a GOSUB a few times. A little experimentation with the value of V will soon establish a satisfactory relationship so that, for example, to move the Robot's vertical axis by the height of one block requires the subroutine to be called 5 times.

At the second level of complexity, you will be able to manipulate the Centronics port directly, and therefore read information as well as write it.

For a Port-mapped Port the BASIC command OUT paddr,1 should be given to turn the motor on. A system with a memory-mapped Port will require POKE paddr,1. Reading the BUSY or ACK line will require some preliminary investigation. The signal may come into the computer on BIT0 of the READ port on the same address as that used for the OUT or POKE commands, but this is by no means a certainty. If the manual doesn't confirm that this is the case, try reading the address you suspect with a short piece of BASIC code:-

```
2000  LET K=IN paddr:REM or use
      =PEEK paddr
2002  PRINT K:GOTO 2000
```

While this code constantly updates X, the voltage on the BUSY/ACK pins should be changed from 0V to +5V. If you observe a change in the number displayed on the screen you will know you have correctly located the port address, and also the number that represents a LOW on the BUSY/ACK lines. However, you'll only be READING the Centronics port if you fit sensors onto the Wall Builder and that introduces the third level of complexity.

SENSING

Two contact switches detect the end-of-travel on the X-axis. These may be sensitive micro-switches of the manufactured variety, or simple strips of springy copper acting as home-made micro-switches. Their positioning is akin to bumpers at the end of a railway line and they are indeed operated by pressure from the body of the moving robot. **Figure 2** shows the construction of the home-made variety. The chain also has two sensors, fitted an equal number of links apart from each other. The 8 stud plate carries one small permanent magnet, and a 3-stud plate carries the other. A blob of rubber solution glue will hold the magnets neatly in place. **Figure 2** indicates the position of the two reed switches which are operated by the magnets. Like the buffers on the X-axis, the Y-axis needs form points of reference which software can detect in order to confirm or adjust the current actual X-Y co-ordinates. Reading the sensors should be done prior to every call to the motor moving sub-routine at 1000. If reading is done from within a LOOP, then an orderly exit from the loop must be anticipated. In the listing starting at line 3000, motor2 is set to travel in reverse until either 20 incremental steps have been made or sensor3 reads zero:-

> **'Reading the sensors should be done prior to every call'**

```
3000  LET N=12:REM 12=0Chex=
      00001100 motor2
3001  LET C=48:REM 48=30hex=
      00110000 sense line3
3002  OUT paddr,C
3004  REM C=48 routes SENSE3 to
      BUSY line
3006  REM now set up for 20 steps
3008  FOR Z=1 TO 20
3010  LET S=IN paddr:REM read
      sense line
3012  IF S=0 THEN LET Z=20:
      GOTO 3018
3013  REM immediate exit if S=0
3016  GOSUB 1000:REM pulse motor
3018  NEXT Z
3020  STOP
```

In the full Wall Building program, you will need to hold the robot's X-Y co-ordinates in two variables, X and Y, and update each variable whenever the appropriate motor moves in a certain direction. A separate counter T should keep a tally of the total amount of movement made by both motors, and when T becomes a high figure, 500 or so, a special reference check should be made to correct any errors in the robot's current position.

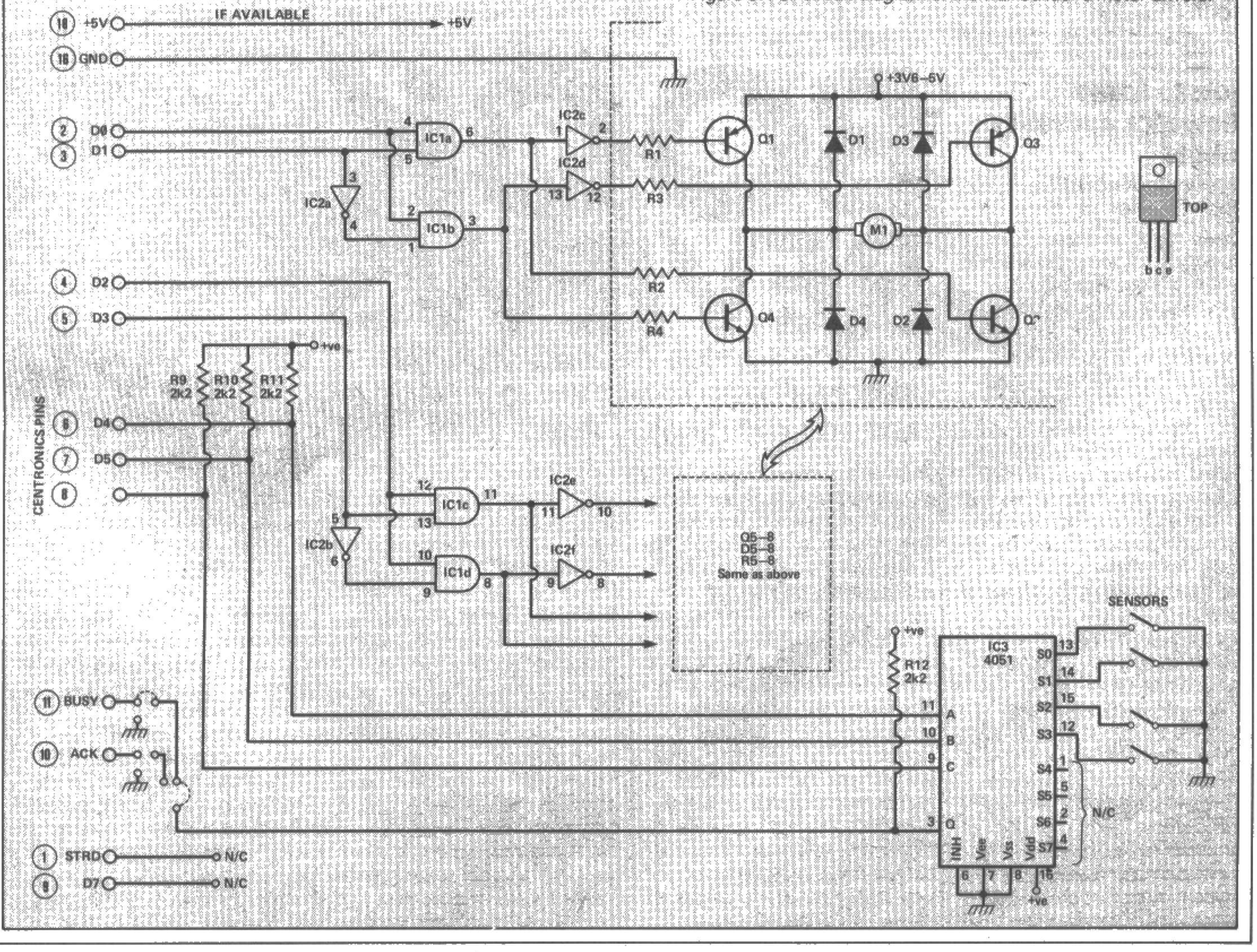
The substance of the checking is shown as "psuedo code" below.

```
LET XO=X:LET YO=Y
LET M equal the X-axis motor
LET C equal the left hand X-axis sensor
OUT paddr,M+C (Drive the motor)
Wait for sensor to equal 0, then output 0.
  (Stop motor)
LET X=0
LET M equal the Y-axis motor
LET C equal the lower Y-axis sensor
OUT paddr,M+C (Drive the motor)
Wait for sensor to equal 0, then output 0.
  (Stop motor)
LET Y=0
Now drive both motors until X=XO and
  Y=YO
LET T=0 Wall Building may now
  continue.
```

MACHINE CODE

The fourth level of complexity is to use machine code to drive the robot. If you decide to build a dedicated CPU to drive your mechanical devices then machine

Figure 5. Full circuit diagram of the wall builder's motor drivers.



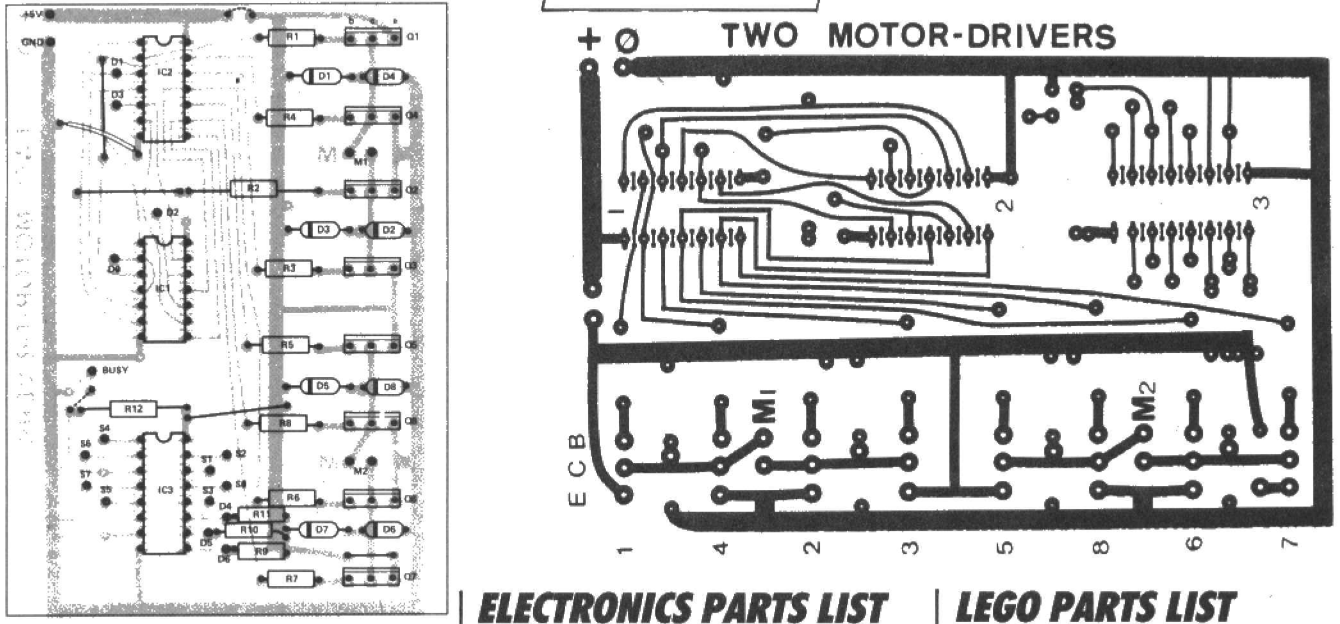


Figure 6. Foil pattern and overlay of the brick builder's driver circuit.

code control will be the only option open to you. Also, if later on you decide that your robots need to do ten things at once and that you have to read a hundred and one sensors rather quickly, you will find that the BASIC high-level language is too slow for the task. Next month we'll be looking at a machine-code driver for Wall Builder, and also building Laurel, a small mouse-like device which can draw graphics on a table-top.

ELECTRONICS PARTS LIST

IC1	
IC2	74LS04 Hex inverter
IC3	4051 CMOS selector or 4051 BE!!
Q1,3,5,7	TIP32
Q2,4,6,8	TIP31
D1-8	1N4148
R1-8	100 ohm 0.25W
R9-12	2.2K 0.25W
2	14pin IC sockets
1	16pin IC socket
3	Duracell or NiCad C-cells
2	small reed switches & magnets
1	battery holder (if required)

LEGO PARTS LIST

10	Track lengths
1	Large studded base plate
2	4.5V Geared motors
4	Flanged wheels
2	Short axles
2	Large gear wheels
2	15-hole beams
1	5-hole beam with peg
3	8-stud narrow plates
3	3-stud narrow plates
Chain links	

Many thanks to Harvey Johns Toyshop, Camden Town for supplying the LEGO.

BOOK REVIEWS

Usborne's range of computer related titles has been extended to include a new volume dealing with robots.

How to make computer controlled robots

By Tony Potter and Chris Oxlade

Those of you who read David Buckley's articles in the first two issues of **Your Robot,** *will remember just how important this experienced robot builder considers wood to be as a material for constructing small hobby robots. The Usborne book provides a design for a computer controlled robot that makes extensive use of balsa wood in a most imaginative way.*

After a brief introduction and hints on how best to use the book, Tony Potter provides a brief guide to the various types of robot systems that are encountered and puts the system described in the rest of the book into context.

At a very early stage, the reader is treated to an illustration of the completed robot showing the relationship of the various elements that go to make up the completed project. The robot is formed by a base plate on which are mounted two independently driven motors that drive a pair of generously proportioned wheels. This base plate provides a mounting point for an arm and gripper arrangement which is one of the most intriguing aspects of the robot. By an ingenious arrangement of pulleys, cord and elastic bands, the gripper on the end of the arm can be made to move through its full range of vertical travel yet at all times remains parallel to the surface on which the robot is operating. The arm and gripper are perhaps the most difficult parts of the project in terms of construction but the copious diagrams explain each stage of the assembly process in such a way that all of the finer points are clearly explained.

At the back of the book are a series of templates that are used to cut out the balsa used to build the robot, and this ensures that all the components are readily fashioned and that everything should fit together perfectly.

In order to achieve computer control of the robot it is obvious that a certain amount of electronics is required and the book describes the necessary relay driver circuit in such a way that even those who have no experience of soldering together circuits should have no problems. There is an excellent guide to the art of soldering that makes full use of an artist's skills to illustrate the do's and don'ts of the art.

Full software is given for a range of popular micros, namely the BBC, Commodore 64, VIC 20 and Spectrum – in the case of the Spectrum it is necessary to supply an I/O board to drive the robot.

When up and running the software allows all aspects of the robot's functions to be controlled by a series of plain language commands entered from the keyboard – UP, DOWN, OPEN, CLOSE etc.

The robot should take about a weekend of work to complete and should cost no more than about £20 to complete or less if you have some of the materials required lying around in a junk box somewhere.

We have seen the robot in operation and after a few initial problems, the beast performed flawlessly managing to to pick up items and move them with deceptive ease.

Usborne are to be congratulated for producing a book that should be of tremendous use both in schools and in the home as it provides an excellent introduction to the concepts of computer control and robotics.

MOTORS MADE SIMPLE

D. S. King rounds off his series on motors with a look at driver circuit techniques.

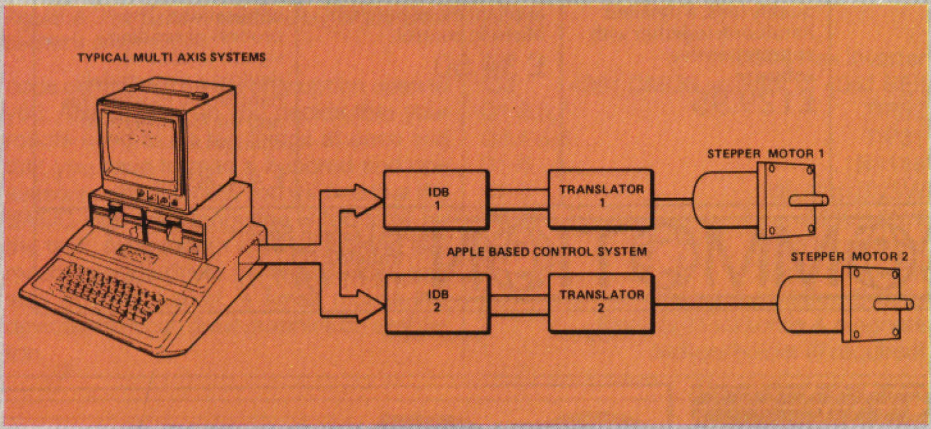


Figure 1 shows how the IBD can be interfaced.

Ideally, an IC type controller should allow command sequences to be stored in its own buffer or execute them as received. Then, from the keyboard, a function oriented, high-level language instruction – such as acceleration, position, or stepping rate should be readily called up using single keystroke operations.

Other functions can be produced from a controller. With sophisticated ICs, a single keystroke will provide stepper location, parameter settings, buffer control data, and mode station information as well.

A specialist in the motor drives field is McLennan Servo Supplies. Amongst the various options in card assemblies, the firm uses Mullard's 4-phase translator IC, (referred to previously) which combines drive logic and output stages to give full step control. With high current variants, the IC provides an attractive logic element for custom-built translator systems.

When increased control over the motor is required, McLennan's IDB microprocessor-based module offers a higher cost solution. But a feature about•this 'intelligent'data buffer is that it can be programmed to compensate for motor backlash.

The IBD can be interfaced with a variety of microcomputers. In one version it has an IEEE-488 interface with 4 I/O control lines and is suitable for Commodore Pet or Hewlett-Packard HP85. When fitted with an ASCII, 8-bit interface, it becomes Apple II compatible, **Figure 1.**

DRIVE CIRCUITS

Having decided upon a method of control, the motor will need a drive circuit to power it. With an electronic drive there are two ways to go, depending upon power level requirements. For peripherals and robotics that need less than about 200W, a linear amplifier, employing series resistors, is the low cost choice. For more power to the motor, a switching amplifier is preferred because of the better efficiency. Both drives utilise transistors.

An 'intelligent' controller, with a suitable power circuit, can be used to drive a number of motors such as may be required for a serial printer, which prints a single character at a time.

This type of machine requires four drives: paper feed, print wheel, shuttle, and ribbon. Although a stepper may be used for all these tasks when printing at slow speed, it is more likely for a shuttle operating above 30 characters that the dc motor will offer better positional accuracy. Similarly, for the print wheel drive, at above 30 characters a stepper tends to overshoot, and oscillation problems limit its use for this duty.

On the robotics side, the expanding personal computer market is encouraging further development of innovative robots. They have already started toddling into the living room, thereby illustrating that these clever machines can walk and avoid obstacles. Most of their functions are motor related and, as robot performance continues to progress, so too will the demand grow for more sophisticated motor-control and drive systems.

TRANSLATORS AND INDEXERS

To round off this overview of motor and controller options, it is pertinent to look briefly at the basic differences in controllers. Mainly, they fall into two general categories: translators (buffered and unbuffered) and preset indexers. Choice is determined chiefly by application, type of computer intelligence available, and the level of programming expertise available.

A translator accepts unbuffered serial data and converts the input data pulses into the proper switching sequence to produce the increment of stepper motion for each input pulse. Using a translator requires the data input source to control the motor's acceleration, maximum slew speed, and deceleration, all by means of varying input pulses.

This dependence on the computer means that all stepper control functions must be designed into the software. The unbuffered translator operates as an almost direct feedthrough from the computer to the motor. But it does have one drawback in that it uses up valuable computer time by limiting the acceptance of usable input data to the motor's operating speed over the range required by the particular task at hand.

PULSES AT HIGH RATE

An alternative is the buffered type of translator. These translators accept pulses at a higher rate than the normal ramp up/down requirements of the motor by initially halving the input pulse frequency being fed to the motor. The translator then gradually increases the frequency to the motor's maximum slew speed until deceleration, when it again halves the input frequency.

LOCAL CONTROL

For greater flexibility a preset indexer offers a wide range of options. For instance, it allows the operator local control of stepper motor programming. This type of controller takes over a number of the computer-based tasks that basic translators depend on, allowing the indexer to operate from a much less sophisticated input. The computer has only to load a timing signal into the indexer, which then takes over.

The indexer's internal oscillator produces the proper pulse signal in a contour matched to the ramp up/down (accelerate/decelerate) setting, which the operator controls with thumbwheel switches on the indexer's front panel.